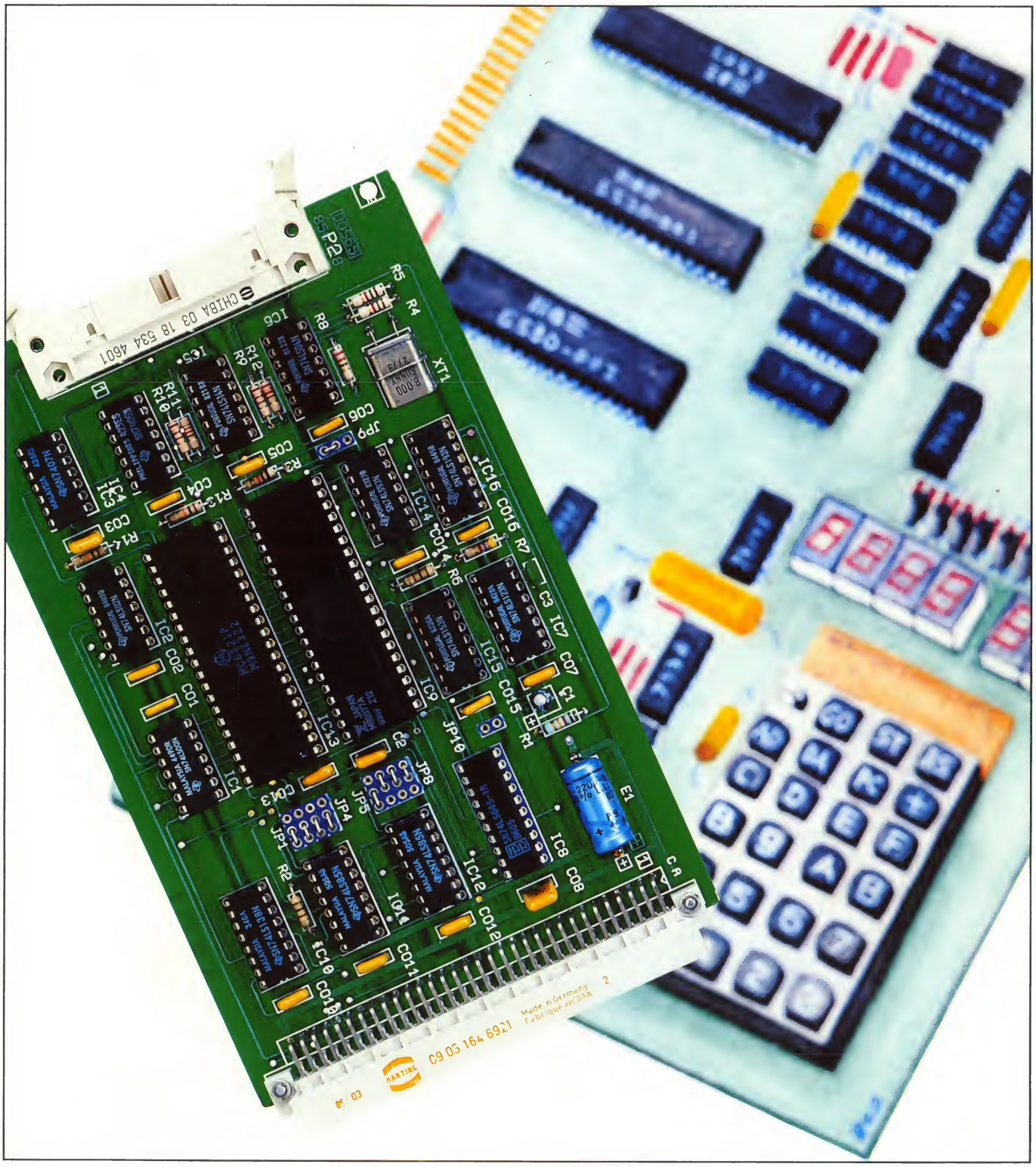
DE 6502 KENNER 57

Twaalfde jaargang nr. 4 augustus 1988



INFORMATIE.

De 6502 Kenner is een uitgave van de KIM Gebruikersclub Nederland. Deze vereniging is volledig onafhankelijk, is statutair opgericht en ingeschreven bij de Kamer van Koophandel en Fabrieken voor Hollands Noorderkwartier te Alkmaar, onder nummer 634305.

Het doel van de vereniging is het bevorderen van de kennisuitwisseling tussen gebruikers van computers die zijn opgebouwd rond een microprocessor uit de 6500-familie. Voorbeelden hiervan zijn onder andere: Elektuur EC-65, Commodore 64, Apple][, Elektuur Junior, Atari 600 en 800.

De eerder genoemde kennisuitwisseling komt onder andere tot stand door 6 maal per jaar de 6502 Kenner te publiceren, door de organisatie van landelijke bijeenkomsten voor de leden, het instandhouden van een softwarebibliotheek op cassette, floppy disk en papier en het beschikbaar stellen van een Bulletin Board.

Landelijke bijeenkomsten:

Deze worden gehouden op bij voorkeur de derde zaterdag van de maanden januari, maart, mei, september en november. De exacte plaats en datum worden steeds in de 6502 Kenner bekend gemaakt in de rubriek Uitnodiging.

Bulletin Board:

Voor het uitwisselen van mededelingen, het stellen en beantwoorden van vragen en de verspreiding van software wordt er door de vereniging een Bulletin Board beschikbaar gesteld. Dit Bulletin Board valt onder de verantwoordelijkheid van één van de bestuursleden en wordt bediend door een zgn. Sysop.

Software Bibliotheek:

Voor het beheer van de Software Bibliotheek streeft het bestuur er naar zgn. Software Coördinators te benoemen. Hierbij wordt gedacht aan een drietal coördinators; één voor EC-65(K) en Junior met OHIO DOS-65D, één voor DOS-65 en één voor diverse andere systemen zoals onder andere Elektuur Junior.

Het Bestuur:

Het bestuur van de vereniging wordt gevormd door een dagelijks bestuur bestaande uit een voorzitter, een secretaris en een penningmeester en een viertal gewone leden.

Voorzitter:
Rinus Vleesch Dubois
Emiliano Zapataplein 2
2033 CB HAARLEM
Telefoon 023-330993

Secretaris:
Gert Klein
Diedenweg 119
6706 CM WAGENINGEN
Telefoon 08370-23646

Penningmeester:
John van Sprang
Tulp 71
2925 EW KRIMPEN A/D IJSSEL

Leden:

Adri Hankel (Bulletin Board)
Willem Kloosstraat 32
7606 BB ALMELO
Telefoon 05490-51151

Erwin Visschedijk
Dillelaan 11
7641 CX WIERDEN
Telefoon: 05496-76764

Gert van Opbroek (Redactie 6502 Kenner)
Bateweg 60
2481 AN WOUBRUGGE
Telefoon 01729-8636

Nico de Vries
Mari Andriessenrade 49
2907 MA CAPELLE A/D IJSSEL
Telefoon 010-4517154

Ereleden:

Naast het bestuur zijn er een aantal ereleden, die zich in het verleden bijzonder verdienstelijk voor de club hebben gemaakt:

Erevoorzitter:

Siep de Vries

Ereleden:

Mevr. H. de Vries van der Winden
Anton Mueller

==========

De 6502 Kenner:

De 6502 Kenner wordt bij verschijnen gratis toegezonden aan alle leden van de KIM Gebruikersclub Nederland. De kopij voor het blad dient bij voorkeur van de leden afkomstig te zijn. Alle kopij wordt door de redactie op bruikbaarheid en publicatiewaarde beoordeeld. Deze twee criteria, in samenhang met de actualiteit, bepalen of en zo ja wanneer het stuk gepubliceerd wordt. De redactie streeft er naar de kopij zoveel mogelijk in zijn oorspronkelijke vorm te plaatsen, Nederlandstalige kopij wordt daarom in principe niet naar een andere taal vertaald. De redactie streeft er naar een Nederlandstalig blad te maken doch het staat de auteur vrij een artikel geheel of gedeeltelijk in een andere taal te schrijven.
Helaas kan de redactie, noch het bestuur, enige aansprakelijkheid aanvaarden voor de toepassing(en) van de gepubliceerde kopij.

Verschijningsdata:

De 6502 Kenner verschijnt op de derde zaterdag van de maanden februari, april, juni, augustus, oktober en december.

Redactie.

De redactie wordt momenteel gevormd door:
Gert van Opbroek

Correspondenten:
Jacques Banser (Sysop)
Bram de Bruine
Antoine Megens
Gerard Reitsma
Rinus Vleesch Dubois

Redactieadres:
Gert van Opbroek
Bateweg 60
2481 AN Woubrugge

INHOUDSOPGAVE

Redactioneel

Het is niet onmogelijk, dat u de 6502 Kenner iets later krijgt dan gebruikelijk. Dit komt doordat ik niet op tijd alle kopij voor het blad bij elkaar had. Normaal gesproken probeer ik altijd het blad in de week na de eerste zaterdag van de maand bij de drukker te krijgen maar nu is dat niet gelukt. Mocht het blad dus niet in de week na de derde zaterdag bij u in de bus liggen, dan vraag ik daarvoor begrip, ik hoop dat het in het vervolg beter gaat.

Ja, het was zeer moeilijk dit blad gevuld te krijgen. Ik voel me persoonlijk ervoor verantwoordelijk te zorgen dat er eens in de twee maanden een 6502 Kenner met een omvang van 48 pagina´s bij de leden ligt en als je dan in totaal slechts ongeveer 15 pagina´s beschikbaar hebt, dan moet je zelf maar aan het schrijven. Nu was ik dat toch al van plan maar toch zou ik het op prijs stellen als u ook wat meer kopij in zou willen dienen, zodat er weer een kopijbuffer ontstaat.

Ondanks het feit dat er niet zo veel kopij binnengekomen is, denk ik toch dat er weer een blad ontstaan is met voor elck wat wilsch. Het blad bevat enkele zeer interessante DOS-65 bijdragen van Antoine Megens. Deze programma´s heeft hij geupload naar het Bulletin Board (053-303902) waarna ik ze er weer vanaf gehaald heb. Het ene programma bevat de DOS-65 game library. Dit is een bibliotheek met een aantal subroutines die zeer goed van pas kunnen komen bij het ontwerpen en implementeren van spelletjes. Het tweede programma bevat enkele scherm-afhandelings routines voor DOS-65 small C.

Voor het eerst sinds lange tijd is er ook iets voor EC-65 beschikbaar. Wij hebben als bestuur momenteel geen enkel inzicht in hoeveel mensen er met bijvoorbeeld EC-65 bezig zijn. Wel willen we toch dit systeem ook blijven ondersteunen en daarom ben ik zeer blij met de bijdrage van Bert van Tiel. Een nieuwtje op EC-65 gebied is misschien dat we bezig zijn met het leggen van contacten met de OSI gebruikersgroep binnen de HCC. Niet dat we opgenomen willen worden in de HCC maar misschien kunnen we als verenigingen toch een vorm van samenwerking bereiken. Ik denk daarbij bijvoorbeeld aan het verspreiden van de wederzijdse software binnen beide verenigingen. Er is contact gelegd met het bestuur van de OSI gebruikersclub en u hoort hier nog meer van.

Verder bevat het blad nog een tweetal grotere bijdragen van mijn hand. Ik hoop dat u ze interessant vindt en dat er niet te gek veel fouten in staan. Het eerste artikel is gebaseerd op een aantal publicaties over RISC-architecturen en gaat over de ontwikkeling en architectuur van moderne processoren, het tweede baseert zich op de Kermit-documentatie en kan gezien worden als een vervolg op het artikel van Gert Klein over het Kermit protocol. Een oud Comal-programma dat door een persbericht opeens weer actueel werd, is ook in dit blad opgenomen; misschien kunt u er iets mee.....

Ja en dan het grote nieuws..... In de laatse bestuursvergadering heeft het bestuur besloten de bakens te gaan verzetten. Wij zien in dat we met de systemen die momenteel de boventoon in de club voeren in de toekomst niet veel verder meer zullen komen en we willen de leden een nieuw beleid voor gaan stellen. Dit nieuwe beleid wordt uit de doeken gedaan in een artikel die ik names het bestuur en als bestuurslid geschreven heb. Het is de bedoeling om dit beleid op de komende ledenvergadering (november in Almelo) aan de leden voor te leggen om dan volgend jaar met dit nieuwe beleid van start te gaan. Het komende najaar willen we gebruiken om al enige invulling aan dit beleid te gaan geven.

Tenslotte nog een leuke anecdote. Vorig jaar in Almelo zag ik van Antoine Megens een spelletje dat mijn aandacht trok. Het was "Bouncing Babies". Het spelletje trok mijn aandacht omdat het na Greedy een van de weinige spelletjes was dat op Junior-achtige systemen draait. Dit spel is door Aintoine ook op het Bulletin Board geplaatst en ik heb zelfs overwogen hem te publiceren. Nu is het volgende gebeurd. Er is, door Antoine?, ook een MS-DOS versie gemaakt en het bestaan van deze versie staat in de HCC-nieuwsbrief van Juli/Augustus (nr. 106). Bovendien staat er in de laatste Computable over dit spel ook een klein verhaaltje (komkomertijd?). Welnu, wilt u dit in de vakpers veelbesproken videospelletje ook eens proberen?, hij staat in DOS-65 formaat en in MS-DOS op ons Bulletin Board en is voor DOS-65 te verkrijgen bij Jan Derksen.

Tenslotte wens ik u voor dit moment nog een mooie nazomer en voor de komende herfst en winter veel (computer-) hobbyplezier.

Uw redacteur,
Gert van Opbroek.

UITNODIGING CLUBBIJEENKOMST

Datum: 17 september 1988
Locatie: Wijkcentrum "De Ringvaart"
Floris van Adrichemlaan 98
2035 VD Haarlem
tel: 023-363856

Entreeprijs: fl. 10,--

Routebeschrijving

AUTO:

Komende uit de richting Utrecht, Amersfoort of Rotterdam:
Afslag Haarlem-Zuid; eerste stoplicht links; bij de tweede kruising met stoplichten links; Floris van Adrichemlaan.

Komende uit de richting Alkmaar:
afslag Haarlem-Zuid; verder zie boven.

OPENBAAR VERVOER:

Vanaf het station Haarlem met buslijn 7, 71, 72 of 77; halte Floris van Adrichemlaan.

Programma:

9:30 Zaal open met koffie
10:15 Opening door de gastheer, onze voorzitter Rinus Vleesch Dubois.
10:30 Voordracht door Adri Hankel:

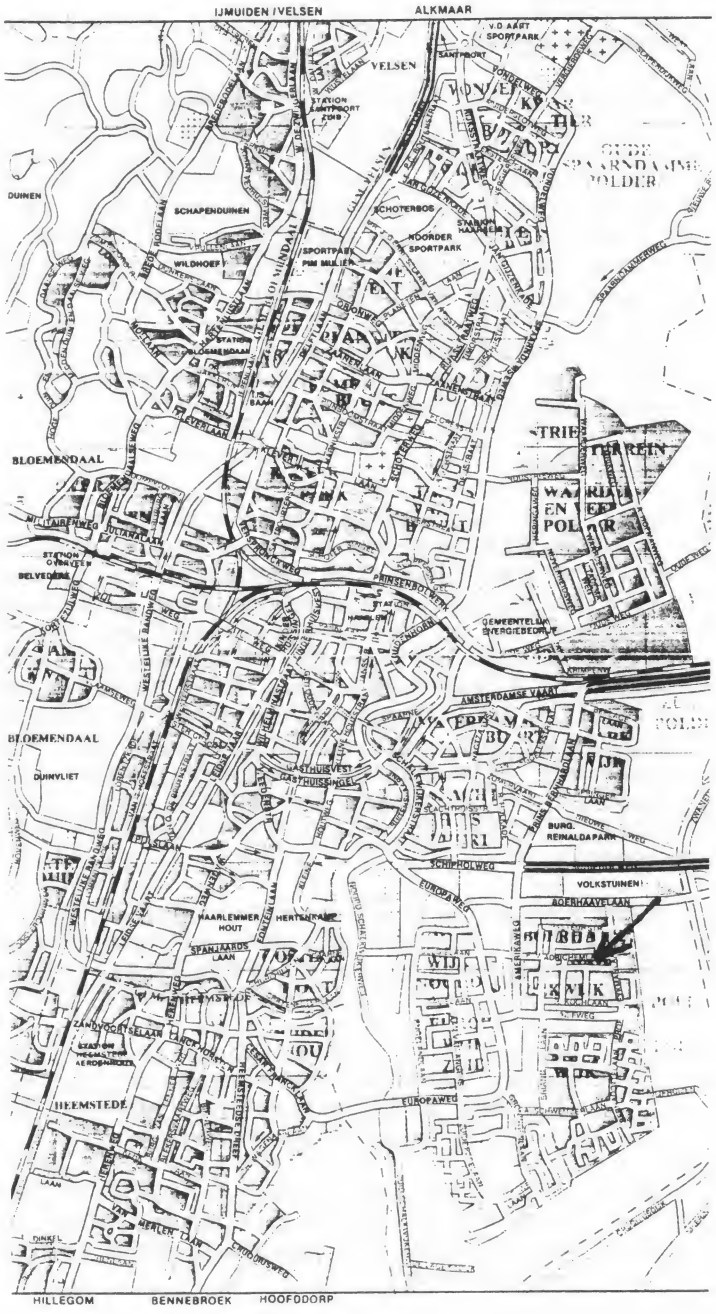
"Single Chip Processoren en de toepassing van een DOS-65 systeem in een professionele omgeving"

Deze voordracht zal gepaard gaan met een demonstratie van lichtkranten

11:30 Forum en markt
12:00 Lunchpauze, Koffie en broodjes op eigen kosten te verkrijgen.

Aansluitend het informele gedeelte bedoeld om kennis, ervaring Public Domain en eigen ontwikkelde software uit te wisselen met uw medeleden. **BRENG DAAROM OOK UW EIGEN SYSTEEM MEE!** (En vergeet de snoeren niet.....)

17:00 Sluiting.



NB. Zoals elders in dit blad beschreven is, wil de club zich ook met andere dan 6502-georienteerde systemen bezig gaan houden. Laat ons daarom ook eens kennismaken met uw systeem, ook al zit er in dit systeem een heel andere processor.

Nieuwe richting voor de KIM Gebruikersclub Nederland

Samenvatting

Op de laatste bestuursvergadering is besloten de leden een nieuw beleid voor te stellen. Het bestuur is van mening dat het noodzakelijk is, de doelstelling van de club, zoals geformuleerd op pagina 2 van de 6502 Kenner te wijzigen. Dit artikel vormt een voorstel tot wijziging van deze doelstelling en alles wat daarmee samenhangt en geeft de achtergronden die tot deze overweging geleid hebben. Het voorstel zal ingebracht worden in de jaarvergadering in november.

Inleiding

Onze club bestaat bijna twaalf jaar onder de naam KIM Gebruikersclub Nederland. In de tijd dat deze club opgericht werd, was de KIM een op een microprocessor gebaseerd systeem dat in de industrie als een soort manusje van alles gebruikt werd. Onze wortels liggen dan ook in de industrie, namelijk bij Forbo Krommenie, de makers van Linoleum. Dit bedrijf toont ook elk jaar nog zijn betrokkenheid bij onze club door ons ieder jaar, in januari, een gastvrij onderdak te verlenen voor een bijeenkomst.

Deze KIM is ook als knutselmachine in de hobby-sfeer terecht gekomen en zodoende is de club ontstaan. De doelstelling van de club is in de statuten geformuleerd als:

1) Bevordering van kennisuitwisseling tussen de gebruikers over de toepassing van de KIM microcomputersystemen en hun eventuele opvolgers.

2) Standaardisering van de te hanteren technieken bij het gebruik en de toepassing van bovengenoemde computersystemen.

Welnu, opvolgers waren er..... In eerste instantie was dat de Junior, een door Elektuur ontwikkeld zelfbouw systeem die vrijwel geheel compatible was met de KIM. Daar de KIM een 6502 microprocessor als hart had, zijn alle computers die deze processor hebben ook bij de club ondergebracht. Dit waren dus Commodores, Apples, Atari´s etc. etc. Verder natuurlijk ook het tweede project van Elektuur rond een 6502 processor: de EC-65.

Ja en toen kwamen er modernere processoren. Er kwam een 65C02, een 6510, een 65816 etc. Strikt genomen kan men systemen met een dergelijke processor nog als opvolger van de KIM beschouwen. Op de markt bleek echter één type systeem er wat marktaandeel beteft met kop en schouders boven uit te steken: De PC-compatibles met een opvolger van de Z80 als processor. Verder zijn er moderne systemen met 68000-processoren gekomen. Op zeker moment is toen gesteld dat de club zich zou gaan richten op alle typen processoren die met een 6 beginnen. En een Apple met Z80 kaart of een EC-65 met een Z80-kaart dan? De 6502 dient als hulpje voor de Z80 en daarom zouden die systemen bij ons thuishoren?

Verder is er nog iets anders gebeurd: Lage prijzen voor de compatibles en PC-Privé. Het zelfbouwen loont niet meer de moeite; wie besteed nog ongeveer duizend gulden om een systeem te bouwen waarvoor dan geen WordPerfect en dBase beschikbaar is. Verder heeft bijna iedereen tegenwoordig op zijn minst toegang tot een PC en aangezien er prachtige software voor die systemen is, wordt het goede oude 6502 systeem niet meer gebruikt. Op zich is dat nog niet zo erg, ware het niet dat hierdoor ons ledental en daardoor de geldmiddelen van de club sterk afnemen. Dit betekent dat als we op die manier doorgaan we binnen enkele jaren doodbloeden door gebrek aan leden en dus geld.

Uitgangspunten

Als we trachten objectief naar de club te kijken, dan zien we een groot aantal positieve punten:

- We hebben een relatief grote harde kern van leden die op een enthousiaste manier met de computerhobby bezig is.

- We hebben een Bulletin Board dat redelijk bezocht wordt en door een enthousiaste Sysop bediend wordt. Verder bevat het BBS software die men zo vaak op andere boards tegenkomt.

- We geven een blad uit dat er naar mijn mening en naar de mening van anderen er best mag zijn en er professioneel uitziet.

- Het kennisniveau van onze leden over hun systemen is hoog tot zeer hoog. Bijna ieder van ons kent zijn systeem door en door en is in staat software te maken van hoge kwaliteit. Het binnen de club ontwikkelde DOS-65 kan men zelfs van professionele kwaliteit noemen.

- Binnen onze club draait men zijn hand er niet voor om zelf diskcontrolers, AD-converters, Operating systemen en Compilers te maken. Kortom de doorsnede van onze club bestaat uit knutselaars.

Als enig nadeel van onze club zie ik het feit dat we met z´n allen halstarrig vast blijven houden aan de 6502. Ik zelf heb dan een op 68000 gebaseerd systeem, dat mag dan nog, maar waarom geen 8088? Omdat er in de KIM een 6502 zat? Of zijn we bang onze identiteit te verliezen? Ik denk niet dat dit zal gaan gebeuren. Feit is wel dat per 1 januari 1988 ons ledental teruggelopen is van bijna 400 naar amper 250. Als iets dergelijks per 1 januari 1989 weer gebeurt, ... Natuurlijk heeft de affaire rond de redactie hierin een bescheiden rol gespeeld maar ik geloof niet dat we nu nog 350 leden zouden hebben als dit niet gebeurd was.

Doelstelling

Om tot een nieuwe doelstelling te komen, is het zinvol te kijken naar de positieve kanten van de club. Deze zijn in de vorige paragraaf geformuleerd.

Naar de mening van het bestuur zou de doelstelling ongeveer als volgt geformuleerd kunnen worden:

1) Het vergaren en verspreiden van kennis over de componenten van microcomputers, de microcomputers zelf en de bijbehorende systeemsoftware.

2) Het stimuleren en ondersteunen van het gebruik van microcomputers in de meer technische toepassingen.

ad. 1.

Opvallend is, dat er niet meer over een type microcomputer of processor gesproken wordt. Dit betekent dus dat we in principe elk type microcomputer kunnen ondersteunen. Bovendien staat er dat we ons bezig zullen houden met systeemsoftware, dit zijn het operatingsysteem, assemblers etc.

ad. 2.

Deze formulering houdt dus in dat we ons bezig zullen houden met technische toepassingen, dus AD-conversie, grafische toepassingen, communicatie, hardware etc. Ook het binnen de club ontwerpen en bouwen van microcomputers behoort tot de mogeljkheden. Dus eigenlijk waar we ons al enige tijd mee bezig houden.

Invulling

Om de genoemde doelstellingen te kunnen bereiken, moeten de volgende dingen gebeuren:

- De statuten moeten gewijzigd worden. Het bestuur zal zich hierover buigen en t.z.t. een voorstel aan de leden voorleggen.

- De naam van het blad moet gewijzigd worden. Dit is in principe een zaak van de redactie. Ik denk dat het zinvol is met ingang van de dertiende jaargang de naam te wijzigen. Als er leuke voorstellen zijn, dan hoor ik ze graag.

- De naam van de club moet misschien gewijzigd worden. Ik zelf vind dat de naam van de club nog best bij de doelstellingen past, maar ik weet dat er leden zijn die er anders over denken. Ook dit is een zaak van het bestuur die met een voorstel naar de leden zal komen.

- We hebben meer kennis nodig. Dit is een soort kip en ei probleem. Heb je geen kennis, dan krijg je geen leden, heb je veel leden dan krijg je kennis. We willen toch proberen, met name middels het blad en het Bulletin Board informatie over andere systemen dan DOS-65 en EC-65 te verspreiden. Dit kan door het publiceren van artikelen over andere systemen. Ik roep daarom ook iedereen op ook artikelen over andere dan 6502-systemen op te sturen. Verder is het zinvol een ledenwerf-actie te starten. Wie wil dat coördineren?

Het bestuur is van mening dat we door deze wijzigingen ook de volgende twaalf jaar een kans van bestaan hebben. Hoewel het voorstel nog niet aan de leden voorgelegd is, willen we toch al beginnen met de invulling van dit beleid. Het bestuur zoekt daarom contactpersonen voor met name Apple (Macintosh), Atari (ST), Commodore (Amiga) en MS-DOS. Verder is kopij over hardware, systeemsoftware en technische toepassingen voor net gelijk welk systeem van harte welkom.

Namens het bestuur:

Gert van Opbroek.

Aanleveren van kopij.

Om de 6502 Kenner te vullen, heeft de redactie **veel** kopij nodig. Aangezien het niet de bedoeling is, dat de redacteur altijd zelf het hele blad volschrijft, doe ik een beroep op de leden. Ik vraag u kopij aan te leveren zodat we deze eventueel in het blad kunnen plaatsen.

Wat komt er zoal in aanmerking voor publicatie? Eigenlijk alles dat iets met de doelstelling van de club te maken heeft. Dat kunnen dus onder andere programmalistings, leuke artikeltjes, hardware etc. zijn, maar ook (opbouwende) kritiek op het blad, de vereniging of andere publicaties in de 6502 Kenner. Ik verzoek u dus allen kopij in te leveren. Hebt u wel een idee, maar gaat het schrijven u minder goed af, dan is de redactie altijd bereid u bij het maken van een publicatie te helpen.

Hoe kunt u de kopij inleveren? In de eerste plaats uiteraard op papier. Wilt u dan wel even op de volgende zaken letten?

- Links en rechts een marge van 1 cm
- Boven een marge van 4 cm en onder een marge van 2 cm
- Verder een zodanige zwart/wit verhouding dat het blad goed door een fotokopieermachine gekopieerd kan worden

Verder kunnen we kopij verwerken die op het bulletin board staat. Als u een kort artikel hebt en een modem bezit, is dit een mogelijkheid. Wilt u dan wel de listings uploaden? Ik kan namelijk voor alleen de Apple of voor CP/M een source verwerken.

Tenslotte schijven, cassettes magneetbanden etc. Verwerkt kunnen worden:

Schijven:

EC-65	5.25" 40 track
DOS-65	5.25", het liefste 40 track, single side
Apple][	5.25" DOS 3.3 en CP/M
OS9	5.25"
Atari	3.5"
MS-DOS	5.25" 360 kb

Cassettes:

Junior/Kim Hypertape, (OCTO)FATE

Magneetband:

VAX-VMS 1600 b.p.i.
Verder eventueel ongelabeld 1600 b.p.i. 9 tracks ASCII en EBCDIC

Tekstverwerkers:

MS-DOS	Wordstar 3.30
	Wordperfect 4.1 en 4.2
CP/M	Wordstar 3.0

DOS-65 Corner

Prijslijst DOS65

- Set manuals bestaande uit:	* Hardware manual * IO65 manual * EDITOR manual * MONITOR manual * ASSEMBLER manual * DOS manual	F 50.-
- 2764 EPROM met IO65		F 35.-
- 2732 EPROM met karakter generator		F 25.-
- Diskette met DOS65 en utilities		F 15.-
- Floppy Disk Controller print		F 50.-
- DOS65 kompleet paket (al het voorgaande) minimaal benodigd om DOS65 te draaien op een (ex) octopus		F 175.-
- Sourcelistings van DOS65, ED, MON, IO65, C compiler		F 25.-
- Sourcelisting van de utilities, AS		F 50.-
- Basic manual		F 25.-
- Small C manual (incl. disk)		F 35.-
- DOS65 entries (incl disk)		F 35.-
- Diverse diskettes met software (zie katalogus)		F 7,50

U kunt uw bestellingen doen bij:

J.D.J. Derksen jr.
DOS65 koordinator
C.P. Soeteliefstraat 41
1785 CC Den Helder
tel: 02230 - 35002 (in het week - end, vraag naar Jan junior)

**** TELEFOON-KLAPPERTJE + ***
*** VOOR DOS-65 2.00 ****

Een sequentieel alternatief.

Wat moet een kleine data-base voor telefoon in zich hebben?
Je moet namen en telefoonnummers kunnen opslaan. Verder is het ook aardig als ook een categoriespecificatie of/en een blokje informatie aan het een naam verbonden zou kunnen worden. Er moeten veel namen in kunnen... Copieeren en corrigeren moet simpel gaan. Men moet er makkelijk en redelijk snel relevante zaken in kunnen zoeken (naam, tel.nr en beroep, aard van een bedrijf ed.). Er moeten wat mogelijkheden zijn, om het overzicht over het bestand te houden. Lijsten moeten gesorteerd zijn. Er moeten op eenvoudige wijze hardcopies van de printer kunnen komen.

De bediening moet simpel zijn. Goed te converteren software (geen peeks en pokes, weinig bijzondere statements ed.).

Het bovenstaande kon vrijwel volledig worden gerealiseerd in het bijgaande telefoonprogramma, ondanks het beperkte geheugen van de Dos-65 computer.

De kern van het probleem nl. was dat geheugen. Een half scherm met een behoorlijke info-vulling vergt gauw 500byte. Een klein, volledig Basic-base-je vergt zelf al gauw 25 a 35kb. En als we dan ook nog een sequentieele lijst van items willen gebruiken dan lopen we vast. En sequentieel werken geeft naast nadelen ook grote voordelen.

Dus heb ik alles maar in stukken geknipt. Bepaalde opdrachten uit een hoofdmenu zijn in enkelvoudige Basicblokken ondergebracht. (Die kunt u nog comprimeren met een evt. nog te publiceren of al gepubliceerd compressor/hernummeringsprogramma of wellicht op een andere manier.) Er zijn vaak zeer veel sprongen gebruikt (voor progr.-structuur niet goed, maar wel compact voor sommige onderdelen en dat telt deze keer).

Ook de lijst, die wij willen maken is in stukken geknipt. Om een lang verhaal kort te maken, de naam Reitsma kan in een - vanuit menu - te definieeren hoofdlijst (vb. ´T´) worden opgelagen in files als TRe.var, waarbij de machine een administratie bijhoudt in ´n file als TR.var. Ter geruststelling, de gebruiker merkt van alle handelingen natuurlijk niets (buiten het opdracht geven, de lijst te creëeren in het hoofdmenu).

Namen zoeken gaat ook via deze weg. Nummers zoeken gebeurt niet vaak, dus dacht ik dat het niet bezwaarlijk zou zijn als dat wat minder snel gaat. Iets dergelijks voor het categorieveld.

De gebruikersvriendelijkheid is tot het extreme opgevoerd. Na het opstarten van Basic en het runnen van Telea, antwoordt u op de eerste vraag (Welke lijst) bv. met ´T´, waarna alles voor zich zelf spreekt. Alles wordt op drive 1 gedaan. Checks worden uitgevoerd, éénletterantwoorden van u J/N ed. worden zonder uw CR uitgevoerd, meerletterantwoorden met CR. In menu-keuze Diversen zit ´Als u het niet meer weet´. Lees dat eerst maar even.

Als u wilt copieeren van de ene lijst naar een andere, moet die lijst er wel zijn. Evt. te creeeren met ´opslagmenagement´ en ´*´ als antwoord op ´naam:´ -> terug naar menu. Zoiets geldt ook voor copy van schijf naar schijf.

Nu kunt u vele honderden namen met bijgaande info op schijf zetten. Zo niet een paar duizend als u weinig info intoetst en 750k schijf-format gebruikt. En snel terugzoeken. Er komen alleen wat veel deelfiles op schijf, maar die verwerkt Dos-65 bij mij tot op heden prima. En mocht u nog meer data kwijt willen, dan kunt u de Basic stukken op de systeemdrive zetten, met een aanpassing van de Dos asn opdracht in de software.

Uw suggesties en raad zijn van harte welkom.

Succes en sterkte bij het foutloos toetsen!!

Overigens: mocht dit in de soep lopen en kan u de zaak niet meer aan de praat krijgen, maak dan uw hoofdlijst weer eens opnieuw met menufunctie opslag na verbeteren van de fout(en).

G.J.Reitsma,
Schouw 19,
1261 LE Blaricum.
Tel. 02152-56645.

Noot van de redactie:
De software staat ophet Bulletin Board of kan op schijf verkregen worden bij de DOS-65 software coördinator.

DOS65 Game Library

Written by A.G.Megens June 1988
Haringvliet 371
8032 HK Zwolle phone: 038-537073

Op het BBS staat een verzameling zeer interessante files:

De DOS65 Game Library

Met behulp van deze files kan men op eenvoudige wijze spelletjes op een DOS65 systeem ontwikkelen.

Mensen die graag de beschikking over de programmatuur willen hebben, kunnen deze downloaden van het Bulletin Board of voor f. 7,50 een floppy bij Jan Derksen bestellen.

De DOS65 game library bestaat uit de volgende files:

JMPVAR.MAC	een JMP table en wat variabele declaraties
SYSCLK.MAC	Om de juiste WAIT loop in te stellen
WAIT.MAC	Om de boel wat te vertragen en HALTKEY scannen
KEY.MAC	Een paar keyboard routines
SCORE.MAC	Om de score bij te houden
SPRITE.MAC	PLOT/ERASE routine voor sprites (nou, ja sprites is een groot woord)
ENDGAME.MAC	Sluit spelletje af, met (nieuwe) high-score check
INSTRUC.MAC	Voorbeeld file hoe instructies erin staan (zie DEMO.MAC)
RANDOM.MAC	Random Number Generator
SCREEN.MAC	Een paar kleine screen routines
SHWKEY.MAC	Voorbeeld file hoe controle toetsen te laten zien (zie ook DEMO.MAC)
DEMO.MAC	Een voorbeeld game dat gebruik maakt van bovenstaande routines

In deze uitgave van de 6502-Kenner wordt de bibliotheek zelf afgedrukt, in het oktober-nummer wordt het voorbeeld-programma DEMO.MAC afgedrukt.

```
**************************************************************************************
*                                                                                    *
*                    G A M E - L I B R A R Y : J M P V A R . M A C                   *
*                                                                                    *
**************************************************************************************

    1   KEYPNT  EQU     $E7B7           ;pointer in inputbuffer
    2   TOUTF   EQU     $E738           ;screen-off flag
    3   TOUTIL  EQU     $E772           ;default screen-off time
    4   TOUTIH  EQU     TOUTIL+1
    5   TOUTL   EQU     TOUTIL+2        ;16 bits screen-off timer
    6   TOUTH   EQU     TOUTIL+3
    7   PTR     EQU     $20             ;zero page general purpose pointer
    8   RUBOUT  EQU     $7F             ;code for RUBOUT key
    9   BELL    EQU     $07             ;code for BELL character
   10   ESC     EQU     $1B             ;code for ESC character
   11   CR      EQU     $0D             ;Carriage Return code
   12   LF      EQU     $0A             ;Line Feed code
   13   TAB     EQU     $09             ;Tab character
   14   FORMF   EQU     $0C             ;Formfeed code
   15   ;
   16   ; jump table DOS65/IO65 routines
   17   ;
   18   OUTCHAR JMP     $F000           ;print character
   19   INKEY   JMP     INPKEY          ;get key without wait
   20   HEXOUT  JMP     $C038           ;print A in hex
   21   PRINT   JMP     $C03B           ;print a string
   22   CLKUPD  JMP     $F5DF           ;software clock to statusline
   23   PUTCUR  JMP     $F024           ;put cursor at screenposition in X,Y
```

```
24   ;-------------------------------------------------------------------------
25   ;       Variable section
26   ;
27   ;       basic set only, add your game dependent variables in main file
28   ;-------------------------------------------------------------------------
29   ;
30   SYSCLK  fcc     0               ;$80/$00 for system clock 1/2 Mhz
31   COUNT   fcc     0               ;char.count
32   PAUSE   fcc     0               ;program pause, initialize at startup!
33   PSAV    fcc     0               ;temp. save PAUSE
34   LIVES   fcc     0               ;# of lives or ships left or, etc...
35   ROUND   fcc     0               ;round counter
36   SCORE   fcc     0,0             ;SCORE
37   HALTF   fcc     0               ;HALT flag, PAUSE key activated!
38   XPOS    fcc     0               ;cursor position
39   YPOS    fcc     0
40   TEMP    fcc     0               ;temp storage
41   SPTR    fcc     0               ;pointer in SPRITE table
42   SPRTAB  fcc     0               ;SPRITE table #1 or #2 flag
43   XSAV    fcc     0               ;save X reg.
44   YSAV    fcc     0               ;save Y reg.
45   ASAV    fcc     0               ;save A reg.
```

```
*****************************************************************************
*                                                                           *
*                   G A M E - L I B R A R Y : S Y S C L K . M A C           *
*                                                                           *
*****************************************************************************
```

```
1    ;=========================================================================
2    ;       CLOCK   let user select system clock
3    ;
4    ;       Input:    none
5    ;       Output:   SYSCLK = $80 - 1 Mhz system clock
6    ;                        = $00 - 2 Mhz
7    ;       Destroys: A,X,Y
8    ;       Calls:    CLS   sends a formfeed to screen
9    ;                 WFKEY wait for key
10   ;                 PRINT print string until NULL (DOS65 routine)
11   ;=========================================================================
12   CLOCK   JSR     CLS
13           JSR     PRINT
14           fcc     'The waitloop in the program is systemclock dependent,'
15           fcc     ' default a 2 Mhz clock',CR
16           fcc     'is presumed, type 1 or 2 to select the proper speed:'
17           fcc     CR,CR,CR
18           fcc     TAB,TAB,'1.......1 Mhz',CR,CR
19           fcc     TAB,TAB,'2.......2 Mhz',CR,CR
20           fcc     'Enter your choise (1/2) > ',0
21   CHOISE  JSR     WFKEY
22           CMP     #'1'            ;<1> selected?
23           BNE     TEST2           ;no check other choise
24           LDA     #$80            ;else put $80 in SYSCLK
25           STA     SYSCLK
26           JSR     PRINT           ;and tell the user
27           fcc     '1      The 1',0
28           JMP     RMESS
29   TEST2   CMP     #'2'            ;<2> selected?
30           BEQ     TWOMHZ          ;yep, then write $00 in SYSCLK
31           CMP     #CR             ;<CR> depressed?
32           BNE     CHOISE          ;no, then illegal choise, ignore it
33   TWOMHZ  LDA     #0              ;else select 2 Mhz system clock
34           STA     SYSCLK
35           JSR     PRINT           ;and tell the user
```

```
 36            fcc     '2       The 2',0
 37   RMESS    JSR     PRINT
 38            fcc     ' Mhz waitloop is now installed',0
 39            RTS

*****************************************************************************************
*                                                                                       *
*                       G A M E - L I B R A R Y : W A I T . M A C                       *
*                                                                                       *
*****************************************************************************************

  1   ;=============================================================================
  2   ;       WAIT    Slow things down.....
  3   ;               Also checks for HALT flag, set by PAUSE key
  4   ;               if set show message on screen on line 1 at pos. 25
  5   ;
  6   ;       Input:    SYSCLK  = $00 - 2 Mhz system clock
  7   ;                         = $80 - 1 Mhz
  8   ;                 SPEED   input speed, controls game speed
  9   ;                 HALTF   flag for PAUSE key
 10   ;                 HALTKEY current HALTKEY value
 11   ;       Output:   message if HALT/PAUSE key is depressed
 12   ;       Destroys: A,X,Y
 13   ;       Calls:    CLKUPD  clock update on status line (I065 routine)
 14   ;                 PRINT   print string until NULL (DOS65 routine)
 15   ;                 INVERS  set display in inverse mode
 16   ;                 NORMAL  set display in normal mode
 17   ;                 SPECIAL show named key
 18   ;                 INKEY   get a key without wait
 19   ;=============================================================================
 20   WAIT    JSR     CLKUPD          ;update system clock and show it
 21           LDX     PAUSE           ;get current speed
 22   WLP1    LDY     SYSCLK          ;get system clock
 23   WLP2    DEY
 24           BNE     WLP2            ;keep counting
 25           DEX
 26           BNE     WLP1
 27           LDX     HALTF           ;HALT flag set?
 28           BEQ     WEXIT           ;no, then done
 29           JSR     PRINT           ;else show message
 30           fcc     $14,25,1,'Press ',0
 31           JSR     INVERS          ;show current HALTKEY in inverse
 32           LDA     HALTKEY
 33           JSR     SPECIAL         ;if named key then show it (e.g. ESC)
 34           JSR     NORMAL
 35           JSR     PRINT           ;second part of message
 36           fcc     ' to continue.....',0
 37   HALT    JSR     INKEY           ;wait for release
 38           BNE     HALT
 39   HALT1   JSR     INKEY           ;wait for HALTKEY
 40           CMP     HALTKEY
 41           BNE     HALT1
 42   HALT2   JSR     INKEY           ;wait for release
 43           BNE     HALT2
 44           LDA     #0              ;clear HALT flag
 45           STA     HALTF
 46           JSR     PRINT           ;clear message
 47           fcc     $14,25,1,'                                        ',0
 48   WEXIT   RTS
```

```
24   ;-------------------------------------------------------------------------
25   ;       Variable section
26   ;
27   ;       basic set only, add your game dependent variables in main file
28   ;-------------------------------------------------------------------------
29   ;
30   SYSCLK  fcc     0               ;$80/$00 for system clock 1/2 Mhz
31   COUNT   fcc     0               ;char.count
32   PAUSE   fcc     0               ;program pause, initialize at startup!
33   PSAV    fcc     0               ;temp. save PAUSE
34   LIVES   fcc     0               ;# of lives or ships left or, etc...
35   ROUND   fcc     0               ;round counter
36   SCORE   fcc     0,0             ;SCORE
37   HALTF   fcc     0               ;HALT flag, PAUSE key activated!
38   XPOS    fcc     0               ;cursor position
39   YPOS    fcc     0
40   TEMP    fcc     0               ;temp storage
41   SPTR    fcc     0               ;pointer in SPRITE table
42   SPRTAB  fcc     0               ;SPRITE table #1 or #2 flag
43   XSAV    fcc     0               ;save X reg.
44   YSAV    fcc     0               ;save Y reg.
45   ASAV    fcc     0               ;save A reg.

*******************************************************************************
*                                                                             *
*                   G A M E - L I B R A R Y : S Y S C L K . M A C             *
*                                                                             *
*******************************************************************************

1    ;=============================================================================
2    ;       CLOCK   let user select system clock
3    ;
4    ;       Input:     none
5    ;       Output:    SYSCLK = $80 - 1 Mhz system clock
6    ;                           = $00 - 2 Mhz
7    ;       Destroys: A,X,Y
8    ;       Calls:     CLS   sends a formfeed to screen
9    ;                  WFKEY wait for key
10   ;                  PRINT print string until NULL (DOS65 routine)
11   ;=============================================================================
12   CLOCK   JSR     CLS
13           JSR     PRINT
14           fcc     'The waitloop in the program is systemclock dependent,'
15           fcc     ' default a 2 Mhz clock',CR
16           fcc     'is presumed, type 1 or 2 to select the proper speed:'
17           fcc     CR,CR,CR
18           fcc     TAB,TAB,'1.........1 Mhz',CR,CR
19           fcc     TAB,TAB,'2.........2 Mhz',CR,CR
20           fcc     'Enter your choise (1/2) > ',0
21   CHOISE  JSR     WFKEY
22           CMP     #'1'            ;<1> selected?
23           BNE     TEST2           ;no check other choise
24           LDA     #$80            ;else put $80 in SYSCLK
25           STA     SYSCLK
26           JSR     PRINT           ;and tell the user
27           fcc     '1       The 1',0
28           JMP     RMESS
29   TEST2   CMP     #'2'            ;<2> selected?
30           BEQ     TWOMHZ          ;yep, then write $00 in SYSCLK
31           CMP     #CR             ;<CR> depressed?
32           BNE     CHOISE          ;no, then illegal choise, ignore it
33   TWOMHZ  LDA     #0              ;else select 2 Mhz system clock
34           STA     SYSCLK
35           JSR     PRINT           ;and tell the user
```

```
 36            fcc     ´2       The 2´,0
 37   RMESS    JSR     PRINT
 38            fcc     ´ Mhz waitloop is now installed´,0
 39            RTS

*****************************************************************************
*                                                                           *
*                    G A M E - L I B R A R Y : W A I T . M A C              *
*                                                                           *
*****************************************************************************

  1   ;=========================================================================
  2   ;       WAIT    Slow things down.....
  3   ;               Also checks for HALT flag, set by PAUSE key
  4   ;               if set show message on screen on line 1 at pos. 25
  5   ;
  6   ;       Input:    SYSCLK  = $00 - 2 Mhz system clock
  7   ;                         = $80 - 1 Mhz
  8   ;                 SPEED   input speed, controls game speed
  9   ;                 HALTF   flag for PAUSE key
 10   ;                 HALTKEY current HALTKEY value
 11   ;       Output:   message if HALT/PAUSE key is depressed
 12   ;       Destroys: A,X,Y
 13   ;       Calls:    CLKUPD  clock update on status line (I065 routine)
 14   ;                 PRINT   print string until NULL (DOS65 routine)
 15   ;                 INVERS  set display in inverse mode
 16   ;                 NORMAL  set display in normal mode
 17   ;                 SPECIAL show named key
 18   ;                 INKEY   get a key without wait
 19   ;=========================================================================
 20   WAIT    JSR     CLKUPD          ;update system clock and show it
 21           LDX     PAUSE           ;get current speed
 22   WLP1    LDY     SYSCLK          ;get system clock
 23   WLP2    DEY
 24           BNE     WLP2            ;keep counting
 25           DEX
 26           BNE     WLP1
 27           LDX     HALTF           ;HALT flag set?
 28           BEQ     WEXIT           ;no, then done
 29           JSR     PRINT           ;else show message
 30           fcc     $14,25,1,´Press ´,0
 31           JSR     INVERS          ;show current HALTKEY in inverse
 32           LDA     HALTKEY
 33           JSR     SPECIAL         ;if named key then show it (e.g. ESC)
 34           JSR     NORMAL
 35           JSR     PRINT           ;second part of message
 36           fcc     ´ to continue.....´,0
 37   HALT    JSR     INKEY           ;wait for release
 38           BNE     HALT
 39   HALT1   JSR     INKEY           ;wait for HALTKEY
 40           CMP     HALTKEY
 41           BNE     HALT1
 42   HALT2   JSR     INKEY           ;wait for release
 43           BNE     HALT2
 44           LDA     #0              ;clear HALT flag
 45           STA     HALTF
 46           JSR     PRINT           ;clear message
 47           fcc     $14,25,1,´                                        ´,0
 48   WEXIT   RTS
```

```
*****************************************************************************
*                                                                           *
*                    G A M E - L I B R A R Y : K E Y . M A C                *
*                                                                           *
*****************************************************************************

    1   ;=========================================================================
    2   ;       INPUT   check for user input on control keys defined at KEYCODE
    3   ;
    4   ;       Input:    KEYCODE table
    5   ;       Output:   jump to selected routine
    6   ;                 also counts DIST for FIRE key
    7   ;                 for QUIT key, returnvalue A<0, else A>0
    8   ;                 keycode in KEY, previous value in PREVKEY
    9   ;       Destroys: A,X,Y,KEYPNT
   10   ;       Calls:    INPKEY get key without wait
   11   ;                 RANDOM generate new RND number
   12   ;
   13   ; Note: Be sure all key routines exit with A>0 or else the program will
   14   ;       act as if the QUIT key was depressed.
   15   ;=========================================================================
   16   INPUT   JSR     INPKEY
   17           BNE     1.f
   18   NOKEY   JSR     RANDOM
   19           LDA     KEY
   20           STA     PREVKEY
   21           LDA     #0              ;A=0, NULL code
   22           STA     KEY
   23           CLC                     ;C=0 , no action taken
   24           RTS
   25   1       LDX     #0              ;check KEYCODE table
   26           STX     KEYPNT          ;ignore rest of inputbuffer
   27   2       CMP     KEYCODE,X
   28           BEQ     3.f             ;found one!
   29           INX
   30           CPX     #NUMKEYS        ;all keys tested?
   31           BNE     2.b             ;no, keep trying
   32           BEQ     NOKEY           ;else act as if no key depressed
   33   3       LDY     KEY
   34           STY     PREVKEY
   35           STA     KEY
   36           TXA                     ;convert X to pointer
   37           ASLA                    ;in JMP table (2 bytes/key)
   38           TAX
   39           LDA     KEYSUB+1,X      ;get address
   40           STA     PTR+1           ;hi
   41           LDA     KEYSUB,X
   42           STA     PTR             ;lo
   43           JSR     GOTHERE         ;and go there
   44           SEC                     ;C=1, action was taken!!
   45           RTS
   46   GOTHERE JMP     [PTR]           ;PTR filled with routine address
   47   ;=========================================================================
   48   ;       TSTKEY  check for QUIT and HALT keys only
   49   ;
   50   ;       Input:    QUIT and HALTKEY code
   51   ;       Output:   A<0, C=1 for QUIT key
   52   ;                 A=1, C=1 and HALTF set for HALTKEY key
   53   ;                 A=0, C=0 if none depressed
   54   ;       Destroys: A,X,Y
   55   ;       Calls:    INPKEY get key without wait
   56   ;=========================================================================
   57   TSTKEY  JSR     INPKEY
   58           BNE     1.f
```

```
 59  3       LDA     #0
 60          CLC
 61          RTS
 62  1       CMP     QUIT
 63          BNE     2.f
 64  QADDR   LDA     #-1
 65          SEC
 66          RTS
 67  2       CMP     HALTKEY
 68          BNE     3.b
 69  HADDR   LDA     #1
 70          STA     HALTF
 71          SEC
 72          RTS
 73  ;=============================================================================
 74  ;       WFKEY   Wait for key and call RANDOM generator while doing so
 75  ;
 76  ;       Input:    none
 77  ;       Output:   Keycode in A and KEY
 78  ;       Destroys: A
 79  ;       Calls:    RANDOM random number generator
 80  ;                 CLKUPD status line clock update
 81  ;                 INPKEY get key without wait
 82  ;=============================================================================
 83  WFKEY   TXA                     ;save X
 84          PHA
 85          TYA                     ;and Y
 86          PHA
 87  1       JSR     RANDOM          ;do RND while waiting for a key
 88          JSR     CLKUPD          ;keep clock up to date
 89          JSR     INPKEY          ;get key without wait
 90          BEQ     1.b
 91          STA     KEY
 92          PLA
 93          TAY                     ;restore Y
 94          PLA
 95          TAX                     ;and X
 96          LDA     KEY
 97          RTS
 98  KEY     res     1               ;used to store KEY value
 99  PREVKEY res     1               ;used in INPUT for previous key
100  ;=============================================================================
101  ;       WFUSER  Wait for user action to continue (e.g. to read instructions)
102  ;       WFSPACE Wait for space-bar with no message
103  ;
104  ;       Input:    none
105  ;       Output:   message on screen
106  ;       Destroys: A,X,Y
107  ;       Calls:    WFKEY suspend until key is depressed, while calling RND
108  ;                 PRINT print string
109  ;=============================================================================
110  WFUSER  JSR     PRINT
111          fcc     CR,CR,'press space-bar to continue.......',0
112  WFSPACE JSR     WFKEY
113          CMP     #' '
114          BNE     WFSPACE
115          RTS
```

```
  116   ;=============================================================================
  117   ;       INPKEY  get key without wait
  118   ;
  119   ;       Input:    none
  120   ;       Output:   keycode in A or NULL if no key depressed
  121   ;       Destroys: A,X
  122   ;       Calls:    INPX get input from device X (I065 routine)
  123   ;=============================================================================
  124   INPKEY  LDX     #9              ;don't wait for key if none depressed
  125           JMP     $F009           ;I065 routine
  126
  127   ;=============================================================================
  128   ;       DEFINE  Redefine control keys
  129   ;
  130   ;       Input:    current values control keys at label KEYCODE
  131   ;                 can define NUMKEYS keys (both in main file, see DEMO.MAC)
  132   ;       Output:   new values control keys
  133   ;       Destroys: A,X,Y,ASAV,XSAV,YSAV
  134   ;       Calls:    CLS    Clear screen
  135   ;                 PRINT  Print string
  136   ;                 SHWKEY Show values control keys (game dependent routine)
  137   ;                 PUTCUR Put cursor at X,Y
  138   ;                 WFKEY  Wait for a key
  139   ;=============================================================================
  140   DEFINE  JSR     CLS             ;clear screen
  141           JSR     PRINT           ;show help info
  142           fcc     'The  current  keyvalues are showed, if you want to change one o
f them move the',CR
  143           fcc     'arrow with the RETURN key until the desired line is reached the
n press the new',CR
  144           fcc     'key  for  that  function. Remember, some CTRL keys may not be u
sed (^C, ^S, ^Q',CR
  145           fcc     0
  146           JSR     PRINT
  147           fcc     'etc.)  because  they  are already used for special functions by
 the system. If',CR
  148           fcc     'you are satisfied leave with ctrl-A',0
  149           LDY     #10
  150           STY     YSAV
  151   SHWVAL  JSR     SHWKEY          ;show current control keys
  152   NEWKEYS LDY     YSAV            ;show arrow
  153           LDX     #21
  154           JSR     PUTCUR
  155           JSR     PRINT
  156           fcc     '-->',0
  157           JSR     WFKEY           ;get a key
  158           AND     #$7F
  159           JSR     PRINT           ;clear errormessage (if any)
  160           fcc     $14,10,8,'                                                  ',0
  161           CMP     #CR             ;move arrow?
  162           BNE     NEWVAL          ;no, new key value
  163           LDX     #21
  164           LDY     YSAV            ;else move two lines down
  165           JSR     PUTCUR
  166           JSR     PRINT           ;first clear arrow
  167           fcc     '   ',0
  168           LDY     YSAV            ;new Y position arrow
  169           INY
  170           INY
  171           STY     YSAV
  172           LDA     #10             ;starts at line 10
  173           CLC
  174           ADC     #(2*NUMKEYS)
  175           CMP     YSAV            ;wrap around?
```

```
176             BNE     NEWKEYS         ;no, continue loop
177             LDY     #10             ;else first line again
178             STY     YSAV
179             BNE     NEWKEYS
180     NEWVAL  STA     ASAV            ;save key code
181             CMP     #1              ;ctrl-A key?
182             BEQ     DEFEX           ;yes, exit
183             LDA     YSAV            ;convert arrow position
184             SEC                     ;to pointer in keyvalues table
185             SBC     #10             ;started at line 10
186             LSRA                    ;divide by 2 (moved 2 lines/key)
187             TAX                     ;this is pointer in keycode table
188             LDA     ASAV
189             STA     KEYCODE,X       ;store new value
190             JMP     SHWVAL          ;and show it
191     DEFEX   LDX     #0              ;same key for more than one function?
192     SAME    LDY     #0
193             STX     XSAV
194             LDA     KEYCODE,X       ;get keycode
195     SO      CPY     XSAV
196             BEQ     NY              ;same pointer then skip test
197             CMP     KEYCODE,Y       ;same code already used?
198             BEQ     SMESS           ;yes, error message and try again
199     NY      INY
200             CPY     #NUMKEYS
201             BNE     SO
202             INX
203             CPX     #NUMKEYS        ;all keys tested?
204             BNE     SAME            ;no, continue
205             RTS                     ;new keys ok, return to main menu
206     ;
207     SMESS   JSR     PRINT
208             fcc     $14,10,8,´No double defined key allowed, please try again´,BELL,
0
209             JMP     SHWVAL
210
211     ;=============================================================================
212     ;       SPECIAL convert special keys to a named string
213     ;               example: ASCII $1B will translate to string ESC
214     ;
215     ;       Input:    key code in A
216     ;                 SPKEYS table
217     ;                 KNAME table
218     ;                 KEYS table
219     ;       Output:   string or char.
220     ;       Destroys: A,X
221     ;       Calls:    PRINT   print a string
222     ;                 OUTCHAR print a character
223     ;=============================================================================
224     SPECIAL LDX     #0
225     TSKEYS  CMP     SPKEYS,X        ;special key?
226             BEQ     PKNAME          ;yes, print its name
227             INX
228             CPX     #SPEND-SPKEYS
229             BNE     TSKEYS
230             CMP     #32             ;control key?
231             BCC     PRCTRL          ;yes
232             JMP     OUTCHAR         ;else just print keycharacter
233     PRCTRL  JSR     PRINT           ;control key starts with ´CTRL-´ string
234             fcc     ´CTRL-´,0
235             CLC                     ;convert code 0..32 to printable ASCII
236             ADC     #´@´
237             JMP     OUTCHAR
238     PKNAME  LDA     KNAME,X         ;get keyname pointer
```

```
239             TAX
240     PRKNAME LDA     KEYS,X          ;and print string found there
241             BEQ     OK
242             JSR     OUTCHAR
243             INX
244             BNE     PRKNAME
245     OK      RTS
246     ;-----------------------------
247     ; Table section
248     ;-----------------------------
249     SPKEYS  fcc ESC,' ',LF,RUBOUT,TAB
250     SPEND   equ $
251     KNAME   fcc 0
252             fcc SPCNAM-KEYS
253             fcc LFNAME-KEYS
254             fcc RUBNAM-KEYS
255             fcc TABNAM-KEYS
256     ;
257     KEYS    fcc 'ESC',0
258     SPCNAM  fcc 'SPACE',0
259     LFNAME  fcc 'LINE-FEED',0
260     RUBNAM  fcc 'RUB-OUT',0
261     TABNAM  fcc 'TAB',0
262     ;
```

```
*****************************************************************************************
*                                                                                       *
*                       G A M E - L I B R A R Y : S C O R E . M A C                     *
*                                                                                       *
*****************************************************************************************
```

```
1     ;=============================================================================
2     ;       HITABLE Show HISCORE table
3     ;
4     ;       Input:    NAMES and HISCORE table
5     ;       Output:   table on screen
6     ;       Destroys: A,X,Y
7     ;       Calls:    PRINT   output string until NULL
8     ;                 OUTCHAR output single char in A
9     ;                 HEXOUT  print A in hexadecimal
10    ;=============================================================================
11    HITABLE JSR     PRINT           ;show header
12            fcc     TAB,TAB,TAB
13            fcc     ESC,'Fpxxxxxxxxxxxxxxxxxxxxxxxxv',ESC,'G',CR
14            fcc     TAB,TAB,TAB
15            fcc     ESC,'Fy',ESC,'G    HIGH SCORES TODAY    ',ESC,'Fy',ESC,'G',CR
16            fcc     TAB,TAB,TAB
17            fcc     ESC,'Fqxxxxxxxxxxxxxxxxxxxxxxxxu',ESC,'G',CR,0
18            LDX     #0
19            LDY     #0
20    HILOOP  JSR     PRINT           ;show rest of cadre
21            fcc     TAB,TAB,TAB,ESC,'Fy',ESC,'G ',0
22    PRNAME  LDA     NAMES,Y         ;print name
23            BEQ     PRSCORE         ;terminated with a NULL
24            INY
25            JSR     OUTCHAR
26            BNE     PRNAME
27    PRSCORE INY
28            JSR     PRINT           ;print score
29            fcc     '   ',0
30            LDA     HISCORE+1,X     ;hi part first
31            JSR     HEXOUT
32            LDA     HISCORE,X       ;then lo part
33            JSR     HEXOUT
```

```
34          JSR     PRINT           ;and another '0' to make it look better
35          fcc     '0 ',ESC,'Fy',ESC,'G',CR,0
36          INX
37          INX
38          CPX     #10             ;5 names/scores stored
39          BNE     HILOOP
40          JSR     PRINT           ;close cadre
41          fcc     TAB,TAB,TAB
42          fcc     ESC,'Frxxxxxxxxxxxxxxxxxxxxxxxxxt',ESC,'G',CR,0
43          RTS
44  ;=======================================================================
45  ;       UPDSCOR Update SCORE, add contents of A to SCORE and refresh the
46  ;               game status line.
47  ;
48  ;       Input:    incremental value in A
49  ;                 ROUND   = current game round
50  ;                 SCORE   = current score
51  ;                 HISCORE = high score
52  ;                 LIVES   = number of lives left
53  ;       Output:   new status line
54  ;       Destroys: A,X,Y
55  ;       Calls:    INVERS set screen inverse mode
56  ;                 NORMAL set screen normal mode
57  ;                 SHOWA  show A in hex. at screen loc. X,Y
58  ;=======================================================================
59  UPDSCOR SEI                     ;disable interrupts (decimal mode)
60          SED
61          CLC
62          ADC     SCORE           ;add A to score
63          STA     SCORE           ;new score
64          BCC     NN
65          LDA     SCORE+1
66          ADC     #0
67          STA     SCORE+1
68  NN      CLD
69          CLI
70  ;=======================================================================
71  ;       UPDATE  Routine for GAME STATUS line
72  ;               (change fields according to current game)
73  ;=======================================================================
74  UPDATE  JSR     INVERS          ;update status line
75          LDY     #25             ;at line 25
76          LDA     ROUND           ;round counter
77          LDX     #25             ;position 25
78          JSR     SHOWA
79          LDX     #59             ;position 59
80          LDA     HISCORE+1       ;hiscore
81          JSR     SHOWA
82          LDA     HISCORE
83          JSR     SHOWA
84          LDX     #38             ;position 38
85          LDA     SCORE+1         ;current score
86          JSR     SHOWA
87          LDA     SCORE
88          JSR     SHOWA
89          LDX     #76             ;position 76
90          LDA     LIVES           ;lives
91          JSR     SHOWA
92          JMP     NORMAL
```

```
 93   ;=============================================================================
 94   ;       SHOWA   Print A in hex on screenpostion X,Y
 95   ;
 96   ;       Input:    A value
 97   ;                 X = X position (column)
 98   ;                 Y = Y position (line)
 99   ;       Output:   A in hexadecimal on screen
100   ;       Destroys: A,X
101   ;       Calls:    OUTCHAR print char. in A
102   ;                 PUTCUR  put cursor at X,Y
103   ;=============================================================================
104   SHOWA   PHA                     ;save value
105           LSRA                    ;get high nibble
106           LSRA
107           LSRA
108           LSRA
109           JSR     PRAC            ;and show it
110           PLA                     ;restore value
111   PRNIB   AND     #$F             ;get low nibble
112   PRAC    SED                     ;enter decimal mode
113           CLC                     ;C=0
114           ADC     #$90            ;convert nibble to ASCII
115           ADC     #$40
116           CLD                     ;clear decimal mode
117           JSR     PUTCUR          ;goto screen position
118           JSR     OUTCHAR         ;and put char. there
119           INX                     ;increment X for next call
120           RTS
```

```
*****************************************************************************
*                                                                           *
*                   G A M E - L I B R A R Y : S P R I T E . M A C           *
*                                                                           *
*****************************************************************************
```

```
  1   ;=============================================================================
  2   ;       PLOT    Plot sprite X
  3   ;
  4   ;       Input   : X contains offset in sprite table SORT
  5   ;                 XPOS, YPOS contain upper left coordinate of sprite.
  6   ;                 sprites are stored in SPRITE and SPRITE2
  7   ;
  8   ;       Output  : sprite displayed on screen
  9   ;       Destroys: A,X,Y,TEMP,SPRTAB,SPTR
 10   ;
 11   ;=============================================================================
 12   PLOT    LDY     YPOS            ;save Y-pos.
 13           STY     TEMP
 14           LDA     #0
 15           STA     SPRTAB          ;default take SPRITE table 1
 16           CPX     #SORT2-SORT     ;which SPRITE table?
 17           BCC     POK             ;#1
 18           INC     SPRTAB          ;else take #2
 19   POK     LDA     SORT,X          ;get figure to draw
 20           STA     SPTR            ;pointer to string
 21   PLOOP   JSR     GOTOXY          ;goto screenposition
 22   GETCHAR LDX     SPTR            ;get stringpointer
 23           INC     SPTR            ;increment stringpointer
 24           LDA     SPRTAB          ;table #1 or #2?
 25           BNE     N2              ;#2
 26           LDA     SPRITE,X        ;get character from string
 27           JMP     P1
 28   N2      LDA     SPRITE2,X
 29   P1      BEQ     DONE            ;NULL means end of data
```

```
30              CMP     #1
31              BEQ     NEXTLIN         ;^A means next line
32              JSR     OUTCHAR         ;else print char.
33              BNE     GETCHAR         ;and get next
34      NEXTLIN INC     YPOS            ;incr. Y-pos.
35              BNE     PLOOP           ;and get next char.
36      DONE    LDA     TEMP            ;restore Y-pos.
37              STA     YPOS
38              RTS
39      ;=======================================================================
40      ;       ERASE   Erase sprite X
41      ;
42      ;       Input   : X contains offset in sprite table SORT
43      ;                 XPOS, YPOS contain upper left coordinate of sprite.
44      ;                 sprites are stored in SPRITE and SPRITE2
45      ;
46      ;       Output  : sprite displayed on screen
47      ;       Destroys: A,X,Y,TEMP,SPRTAB,SPTR
48      ;
49      ;=======================================================================
50      ERASE   LDY     YPOS            ;save Y-pos.
51              STY     TEMP
52              LDA     #0
53              STA     SPRTAB          ;default take SPRITE table 1
54              CPX     #SORT2-SORT     ;which SPRITE table?
55              BCC     EOK             ;#1
56              INC     SPRTAB          ;else take #2
57      EOK     LDA     SORT,X          ;get figure to erase
58              STA     SPTR            ;pointer to string
59      ELOOP   JSR     GOTOXY          ;goto screenposition
60      ERACHAR LDX     SPTR            ;get pointer
61              INC     SPTR            ;incr. pointer
62              LDA     SPRTAB          ;table #1 or #2?
63              BNE     NR2             ;#2
64              LDA     SPRITE,X        ;get character from string
65              JMP     E1
66      NR2     LDA     SPRITE2,X
67      E1      BEQ     DONE            ;NULL means end of data
68              CMP     #1              ;next line?
69              BEQ     NXTLINE
70              CMP     #ESC            ;ESC sequence?
71              BNE     NOESC           ;no
72              INC     SPTR            ;yes, then ignore next char.
73              BNE     ERACHAR         ;and get next
74      NOESC   LDA     #' '            ;else erase char.
75              JSR     OUTCHAR
76              BNE     ERACHAR         ;until end
77      NXTLINE INC     YPOS
78              BNE     ELOOP
79      ;
80      ; Note: Table SPRITE, SPRITE2 and SORT are to be inserted in MAIN program
81      ;       (also see DEMO.MAC file)
```

```
*****************************************************************************
*                                                                           *
*                   G A M E - L I B R A R Y : E N D G A M E . M A C         *
*                                                                           *
*****************************************************************************
```

```
1       ;==================================
2       ; End of game, check HISCORE etc.
3       ;==================================
4       ENDGAME LDX     #0
5       SCHECK  LDA     SCORE+1
```

```
    6             CMP     HISCORE+1,X     ;greater or equal than hiscore?
    7             BEQ     SWAP            ;equal, then also test low byte
    8             BCS     NEWHIGH         ;yep, new high score!
    9   NSCORE    INX
   10             INX
   11             CPX     #10             ;all hiscores tested?
   12             BNE     SCHECK
   13             JMP     QUEST           ;then ask for another game
   14   SWAP      LDA     SCORE           ;get low byte score
   15             CMP     HISCORE,X
   16             BCC     NSCORE          ;smaller then hiscore, try next
   17   NEWHIGH   STX     XSAV
   18             CPX     #8              ;last high score?
   19             BEQ     SSCORE          ;yes, then doesn't need inserting
   20             LDY     #7              ;make room for new high score
   21   INSERT    LDA     HISCORE,Y
   22             STA     HISCORE+2,Y
   23             CPY     XSAV
   24             BEQ     SSCORE
   25             DEY
   26             JMP     INSERT
   27   SSCORE    LDA     SCORE
   28             STA     HISCORE,X       ;store new high score in table
   29             LDA     SCORE+1
   30             STA     HISCORE+1,X
   31             TXA                     ;X=0,2,4,6,8
   32             ASLA                    ;*2
   33             ASLA                    ;*4
   34             ASLA                    ;*8
   35             STA     TEMP            ;A=0,16,32,48,64
   36             CPX     #8
   37             BEQ     OVWRITE         ;last name doesn't need inserting
   38             LDY     #63             ;start with second last name
   39   MKROOM    LDA     NAMES,Y         ;overwrite it with previous name
   40             STA     NAMES+16,Y
   41             CPY     TEMP
   42             BEQ     OVWRITE
   43             DEY                     ;until new high score
   44             JMP     MKROOM
   45   OVWRITE   LDX     #0              ;clear this name
   46             LDA     #' '
   47             LDY     TEMP
   48   CLRNAM    STA     NAMES,Y
   49             INY
   50             INX
   51             CPX     #15
   52             BNE     CLRNAM
   53             JSR     PRINT
   54             fcc     $14,17,1,'Congratulations, you may enter your name in the HISCOR
E'
   55             fcc     $14,17,2,'table. Enter your name (15 char. maximum) end with <CR
>',BELL
   56             fcc     $14,30,4,'...............',$14,1,6,0
   57             JSR     HITABLE
   58             JSR     PRINT
   59             fcc     $14,30,4,0
   60             LDX     #0
   61             STX     COUNT           ;character count
   62             LDY     TEMP            ;table pointer
   63   GNAME     JSR     WFKEY           ;get key
   64             AND     #$7F            ;is it RUBOUT?
   65             CMP     #RUBOUT
   66             BNE     TNEXT           ;no, test if ASCII
   67             LDX     COUNT           ;at start of line?
```

```
68          BEQ     RBELL           ;yes, then ring bell
69          DEY                     ;else clear previous character
70          DEC     COUNT
71          LDA     #' '
72          STA     NAMES,Y         ;in table
73          JSR     PRINT
74          fcc     8,'.',8,0       ;also on screen
75          JMP     GNAME
76  RBELL   LDA     #BELL           ;ring bell
77          JSR     OUTCHAR         ;when at start of line
78          JMP     GNAME           ;or buffer full
79  TNEXT   CMP     #CR             ;RETURN entered?
80          BEQ     QUEST           ;yes, exit name input
81          CMP     #32             ;control key?
82          BCC     GNAME           ;yes, ignore it
83          LDX     COUNT
84          CPX     #15             ;buffer full?
85          BNE     SCHAR           ;no, continue
86          JMP     RBELL           ;ring bell to warn user
87  SCHAR   STA     NAMES,Y         ;store character in table
88          JSR     OUTCHAR         ;and print it
89          INY                     ;incr. both pointer and counter
90          INC     COUNT
91          JMP     GNAME
92  ;==============================
93  ; Ask for another game
94  ;==============================
95  QUEST   JSR     PRINT
96          fcc     $14,1,6,0
97          JSR     HITABLE         ;show new name in table
98          JSR     PRINT
99          fcc     $14,25,4,'Do you want to play again (y/n) ? ',BELL,0
100         JSR     WFKEY
101         AND     #%01011111
102         CMP     #'N'
103         BEQ     DOS
104         CMP     #'Y'
105         BNE     QUEST           ;error in input
106         JMP     START
107 ;=============================================================================
108 ; Because the KEYIN without wait is used, the screen-off timer is updated, but
109 ; not executed, the result is that the screen switches off when you've played
110 ; more than DPTIME seconds and return to DOS. To prevent this the TOUTx timer
111 ; is reset to default before exit.
112 ;=============================================================================
113 DOS     LDA     TOUTIL          ;reset screen-off timer
114         LDX     TOUTIH
115         STA     TOUTL
116         STX     TOUTH
117         LDA     #$FF
118         STA     TOUTF
119         JMP     CLS             ;clear screen and return to DOS65
120 ;
```

```
*****************************************************************************
*                                                                           *
*                    G A M E - L I B R A R Y : I N S T R U C . M A C        *
*                                                                           *
*****************************************************************************

    1   ; GAME dependent, include in main and change it there (see DEMO.MAC)
    2   ;
    3   ;=========================================================================
    4   ;       INSTRUC show instructions and control keys on screen
    5   ;
    6   ;       Input:    None
    7   ;       Output:   Instructions
    8   ;       Destroys: A,X,Y
    9   ;       Calls:    SHWKEY show control keys
   10   ;                 WFUSER wait for space-bar with message (KEY.MAC file)
   11   ;                 CLS    clear screen
   12   ;=========================================================================
   13   INSTRUC JSR     CLS             ;clear screen
   14           JSR     PRINT
   15           fcc     '<put instructions here>',0
   16           JSR     SHWKEY          ;show control keys
   17   RTITLE  JSR     WFUSER
   18           RTS

*****************************************************************************
*                                                                           *
*                    G A M E - L I B R A R Y : R A N D O M . M A C          *
*                                                                           *
*****************************************************************************

    1   ;=========================================================================
    2   ;       RANDOM  simulates a random number generator using a 24 bits shift
    3   ;               register.
    4   ;               N.B. at least one of the RND bytes should be initialized
    5   ;                    before RANDOM is called (any number <> 0 will do)
    6   ;
    7   ;       Input:   RND,RND+1,RND+2
    8   ;       Output:  A (8 bits RND number)
    9   ;       Destoys: A
   10   ;       Calls:   none
   11   ;=========================================================================
   12   RANDOM  LDA     RND             ;simulate random numbers
   13           AND     #$48            ;with 24 bits shiftregister
   14           ADC     #$38
   15           ASLA
   16           ASLA
   17           ROL     RND
   18           ROL     RND+1
   19           ROL     RND+2
   20           LDA     RND
   21           RTS
   22
   23   RND     res     3,1
   24
```

(Wordt vervolgd)

NB. De listings zijn niet met de assembler geproduceerd maar met tekstverwerkers. De getoonde layout kan dus niet met een assembler-switch ingesteld worden.

Katalogus DOS65 software

DOS65 extra 1.0 div utilities, spelen en C compiler (zonder handl)	15.-
ISO pascal compiler inklusief dokumentatie	25.-
Small C compiler incl manual op papier	35.-
FORTH inklusief editor, assembler en dokumentatie	15.-
DOS65 entries disk + manual op papier	35,-
Astrid datakomunikatie pakket inklusief viditel (3 disks)	25.-
Kermit datakomunikatie pakket inklusief dokumentatie	25.-

Uit de volgende files kunt u zelf uw programma's uitzoeken.
Prijs F 7.50 per schijf

SUB.ASC	Basicode-2 subroutines voor DOS65 basic 2.00
DOBBEL.BAS	Dobbelen met je computer
FINAN.BAS	Financieel programma
HANOI.BAS	De torens van Hanoi
HHBOEK.BAS	Houdt hiermee uw huishouding bij
YATSE.BAS	Het bekende spel yatse met de computer
STARTREK.BAS	Speelt de rol van captain Kirk en vernietig de klinkonen
KLOK.BAS	Maakt de tijd groot zichtbaar
WORLDTI1.BAS	Vraagt de tijd op van bijvoorbeeld China
KILLMAN.BAS	Psychologische test
SORTER.BAS	Sorteert A,B,C en D's
OTHELLO.BAS	Speelt othello tegen de computer
LETPUZ.BAS	Stelt zelf je speurpuzzel samen
BRUTONET.NOS	Berekening nettoloon in basicode-2
FRUITMACH.2	Spelletje in basicode-2
NUMWIS.NOS	Integreren, differentieren in basicode-2
WARGAMES.BAS	Laat je hayesmodem modems zoeken (GWBASIC)
PLOT.DEMO	Grafisch scherm voor DOS65 met 6845 VDU en viditel char.gen.
TELE	Telefoonklapper pakket met dokomentatie
BABIES	Een videospelletje voor DOS65
BABIES.DOC	Uitleg bij babies
STOWAWAY	Een videospelletje voor DOS65
STOW.DOC	Uitleg bij stowaway
USURPATOR	Schaakprogramma
USUR.DOC	Handleiding voor usurpator
NORGRAPH	Cansel graphic & inverse mode
SEARCH	Zoekt in ASCII files naar strings
CRTC	Utility voor het veranderen van de registers van de 6845
SETCRTC	Zet de veranderde registers opnieuw in de 6845
GDP	Tekenprogrammatje voor de grafische kaart van Elektuur
TIMEDATE	Vraagt om de tijd een datum (voor in login.com)
BASICO.D65	Toelichting basicode-2 met DOS65
SCRED52	Screeneditor voor DOS65 basic V2.00
TAPE	Basicode-2 lees/schrijf programma
LTAB	Load tab posities voor ED
STAB	Save tab posities voor ED
RR	Renumber utility voor basic programma's zie 6502 kenner 56

Deze lijst wordt steeds bijgewerkt. Voor de meest update lijst met software kunt u het beste op het bulletinboard kijken. U kunt de software daar downloaden of besteilen bij de DOS65 coordinator. Indien u geen modem heeft dan kunt u een lijst aanvragen bij de DOS65 coordinator (wel even een envelop met postzegel bijsluiten a.u.b.)

VIDEO ROUTINES VOOR DOS-65.

De onderstaande is een zogenaamde "HEADER"-file voor de DOS-65 Small C-compiler. Met deze file kan men in DOS-65 een aantal video-routines aanroepen.

Het bijbehorende demo programma (DEMO.C) geeft enkele voorbeelden voor het gebruik van de routines.

De twee files zijn gedownload vanaf het bulletin board (053-303902).

```
/*****************************************************************/
/*                                                               */
/*      File: video.h                                            */
/*---------------------------------------------------------------*/
/*      Small C video control routines for I065/DOS65 system     */
/*              written by A.Megens  spring 1987                 */
/*                                                               */
/*****************************************************************/

cls()
{ printf("\014"); }      /* formfeed */

invers()
{ printf("\033i"); }     /* ESC i */

normal()
{ printf("\033n"); }     /* ESC n */

grafon()
{ printf("\033F"); }     /* ESC F */

grafoff()
{ printf("\033G"); }     /* ESC G */

bell()
{ printf("\007"); }      /* BELL */

home()
{ printf("\034"); }      /* HOME */

crlf()
{ printf("\n\r"); }      /* CRLF */

xcrlf(n)                 /* multiple CRLF */
int n;
{ while (n--) printf("\n\r"); }


/*****************************************************************************/
/*                                                                           */
/* gotoxy(x,y)  - cursor to coordinate x,y. The function is aborted with     */
/*                return value 0 when x or y are out of range.               */
/*                                                                           */
/*****************************************************************************/

gotoxy(x,y)
char x,y;
{ if ((x<0)|(x>79)|(y<0)|(y>25)) return(0);   /* if error return 0, abort */

  /*-----------------------------------------------------------------------*/
  /* due to a bug in the direct cursor addressing mechanism with ^T ($14) */
  /* (error in DOS65 when X or Y is 13 (CR)) here assembly code is used,  */
  /* instead of the much simpler:                                          */
```

```
  /*                                    printf("\024%c%c",x,y);          */
  /*-------------------------------------------------------------------*/

  #asm
        ldc.u 0         ;get first argument (small C macro)
        pha             ;save on stack
        ldc.u 2         ;get second argument
        tay             ;in Y-reg.
        pla             ;get stack-value
        tax             ;in X-reg.
        jsr $F024       ;I065 routine (goto screen coordinate X,Y)
  #endasm

  return(1); /* no errors, return 1 */
}

/*********************************************************************/
/*                                                                   */
/* square(x,y,width,height)     - draw a square width,height with upper */
/*                                left corner at coordinate x,y      */
/* The function is aborted with return value 0 when the square does not */
/* fit on the screen. Else return value is 1.                        */
/*                                                                   */
/*********************************************************************/

square(x,y,width,height)
int x,y,width,height;
{ int i;
  if ((x<1)|(width<1)|((x+width)>79)|(y<1)|(height<1)|((y+height)>24))
        return(0);  /* return 0 if error in arguments, function aborted */

  grafon();gotoxy(x,y);
  printf("P");for (i=1;i<width;i++) printf("X");        /* draw topline */
  printf("V");
  for (i=1;i<height;i++)
  {     gotoxy(x,y+i);printf("Y");                     /* draw sides */
        gotoxy(x+width,y+i);printf("Y");
  }
  gotoxy(x,y+height);printf("R");
  for (i=1;i<width;i++) printf("X");                    /* draw bottom */
  printf("T");grafoff();

  return(1); /* no error, return 1 */
}



/*****************************************************************/
/*                                                               */
/*      File: demo.c                                             */
/*---------------------------------------------------------------*/
/*              Demo program for video routines in video.h       */
/*                                                               */
/*****************************************************************/

#include video.h        /* get library functions */

main()
{ int i;
  char c;

  cls();
  invers();printf("\n\t\t\tDEMO SMALL C VIDEO ROUTINES\n\n");normal();
```

```
  printf("\tNew routines: cls(), invers(), normal(), grafon(), grafoff(),");
  printf("home(),\n");
  printf("\tbell(), crlf(), xcrlf(n), gotoxy(x,y), square(x,y,width,height).");
  xcrlf(2);
  printf("\tnormal mode: @ A B C D E F ^ ? P Q R S T U V W X Y Z\n");
  printf("\tgrafic mode: ");
  grafon();printf("@ A B C D E F ^ ? P Q R S T U V W X Y Z\n");grafoff();

  square(1,1,78,23); /* maximum width and height for square function */
  square(65,7,1,1);  /* smallest possible square (width=1, height=1) */
  square(5,6,65,3);

  /* square is not drawn when it does not fit on screen */
  i=1;
  while (square(30-3*i,10+i,2*i+2,i+1)) i++; /* return value then 0 else 1 */
  i=1;
  while (square(30+3*i,10+i,2*i+2,i+1)) i++;

  grafon();gotoxy(75,21);printf("PWV"); /* grafic signature */
  gotoxy(75,22);printf("QUY");grafoff();

  gotoxy(45,11);printf("Press a key to exit.");
  bell();c=getchar();
  cls();
}
```

==========

Nieuw voor DOS-65: Een Pascal-compiler.

Sinds kort bestaat er voor DOS-65 een nieuwe compiler: ISO-Pascal !

Pascal is een taal die door Niklaus Wirth aan het EHT te Zürich, Zwitserland ontwikkeld is. Hij heeft de taal ontwikkeld om zijn studenten op te leiden in het gestructureerd programmeren. Nadien heeft de taal een grote vlucht genomen. Op bijna alle universiteiten en hogescholen wordt studenten geleerd in deze taal te programmeren. Verder zijn er al diverse commerciele toepassingen in Pascal geprogrammeerd. Een inleidend artikel over Pascal is gepubliceerd in de 6502 Kenner nr. 42.

Sinds enkele jaren bestaat er een ISO-standaard voor Pascal. En sinds kort is er een compiler voor DOS-65 beschikbaar die de volledige ISO-standaard ondersteunt. Dit betekent dat in de taal alle zaken zoals pointers, floating point getallen enzovoort voorkomen.

Aangezien het DOS-65 beleid erop gericht is, ongeveer tegen kostprijs software te verspreiden, kunt u dit fantastische stuk software voor slechts f. 25,-- bij Jan Derksen, de DOS-65 coördinator bestellen. Voor dit (kleine) bedrag krijgt u dan een schijf met daarop de compiler en de documentatie. De documentatie is dus niet op papier verkrijgbaar. Elders in dit blad vindt u een compleet overzicht van wat er verder aan software voor DOS-65 beschikbaar is.

Ik hoop dat de nieuwe Pascal-compiler voor u aanleiding is kopij in deze prachtige taal naar mij op te sturen. Mocht u overigens problemen hebben met het programmeren in Pascal, ook ik heb tijdens mijn opleiding veel met de taal te maken gehad en verder programmeer ik voor mijn werk bijna dagelijks is Pascal. Als u vragen hebt, kunt u dus gerust contact met mij opnemen.

Gert van Opbroek
01729-8636

CISC EN RISC, EEN INLEIDING.

Gert van Opbroek
Bateweg 60
2481 AN Woubrugge (01729-8636)

Inleiding.

In de diverse vaktijdschriften, wordt er veelvuldig melding gemaakt van processoren die gebaseerd zijn op het zogenaamde RISC-concept. In dit artikel wil ik enige aspecten van deze nieuwe ontwikkeling belichten.

In de eerste plaats is het waarschijnlijk zinvol de betekenis van de afkortingen RISC en CISC te verduidelijken.

RISC is een afkorting van **R**educed **I**nstruction **S**et **C**omputer; dus een computer met een verminderd aantal instructies.

CISC staat voor **C**omplex **I**nstruction **S**et **C**omputer, dus een computer met een gecompliceerde set instructies.

Van beide types zijn er in het aanbod van de diverse fabrikanten voorbeelden te vinden. Voorbeelden voor CISC-processoren zijn onder andere de 68x00 en 80x86, dus die processoren die momenteel algemeen toegepast worden. Als voorbeeld voor RISC-processoren kunnen dienen: de ARM (Acorn Risc Machine) die in de Archimedes zit, de door Motorola aangekondigde MC88000, de AM 29000 van AMD en de SPARC van Sun. Ook transputers (processors, speciaal bedoeld voor paralelle verwerking) kunnen we als RISC-processoren opvatten.

Er zijn mensen, die de bekende 6502 opvatten als de eerste RISC-processor. Kijken we naar de architectuur van de 6502, dan zijn er inderdaad motieven voor deze uitspraak te vinden, toen echter de 6502 ontworpen werd, bestond het begrip RISC nog niet eens, men bouwde een processor die met de toen geldende technologie het maximaal haalbare was.

Historische ontwikkeling.

In Byte [1] beschrijft Pete Wilson het ontstaan van de diverse architecturen van microprocessors. Uitgangspunt voor zijn verhaal vormt de stand van de technische ontwikkelingen op het moment dat een bepaalde processor ontwikkeld werd. Al met al een zeer leesbaar verhaal dat voor mij een heel nieuw licht op deze zaak wierp.

In de begintijd van de microprocessor, had men de technologie om een chip met maximaal 10.000 transistors te ontwerpen en te produceren. Met deze 10.000 transistors heeft men een processor gebouwd die zo veel mogelijk aan de toen geldende wensen voldeed. Een tweede gevolg van de eenvoudige technologie was dat geheugen duur was. (De pioniers in onze club weten misschien nog wel wat de 2102, 1024 bits statisch geheugen in een IC, koste). Eén en ander heeft geleid tot microprocessors met eigenschappen zoals de 6502:

- Een minimum aan registers.
- Een adresseerbereik van 64 kb.
- Een 8 bits opcode (want 256 instructies was meer dan genoeg).
- Geen gecompliceerde rekeninstructies zoals vermenigvuldigen en delen.

Om toch nog zoveel mogelijk met de transistoren en het beschikbare geheugen te kunnen doen, heeft, met name de 6502, een aantal zeer slimme eigenschappen:

- Een accumulerend register (ACCU) met LOAD en STORE instructies. Dit heeft als voordeel dat, bijvoorbeeld bij een optelling, slechts één van de twee operanden uit het geheugen gehaald hoeft te worden. Daar het resultaat ook weer in de ACCU terecht komt, spaart dit ook weer een geheugen benadering (memory access) uit.
- Indirecte adressering via pagina 0. Dit betekent dat in het programma slechts een één byte verwijzing naar een locatie binnen de laagste 256 geheugenlocaties staat. Op deze plaats staat dan het volledige twee bytes adres van de operand. Op deze manier kan men op eenvoudige wijze datastructuren benaderen.
- Indexering. Om de zaken nog wat gemakkelijker hanteerbaar te maken, heeft men ook nog een indexering op de indirecte adressering toegepast. Dit zelfs op twee manieren. Bij de eerste, wordt de inhoud van het index-register opgeteld bij de verwijzing naar pagina 0 in het programma (derde plaats vanaf adres 7). De inhoud van deze en de volgende locatie op pagina 0 geeft het adres van de operand. Bij de tweede wordt de inhoud van het index-register opgeteld bij het adres dat op pagina 0 in het geheugen staat (ga naar de locatie 10 bytes van het adres in locatie 3 en 4).

- Omdat men de benadering van pagina 0 middels één byte toch al had, heeft men ook de rechtstreekse adressering van operanden op pagina 0 middels een twee byte instructie geïmplementeerd. Verder heeft men ook een idexering op de absolute en pagina 0 adressering toegestaan.

Er wordt wel eens beweerd, dat de 6502 een processor is met ruim 256 registers. In zekere zin is dit ook wel zo. Mensen die op bijvoorbeeld de PDP-11 gewerkt hebben of op een machine, gebaseerd op de 68000, weten waarschijnlijk wat hiermee bedoeld wordt. Een register kan men opvatten als een zeer snel geheugen. Bovendien kan men over het algemeen met vrij korte instructies een register benaderen. In de praktijk worden dan ook de veel gebruikte variabelen en pointers naar datastructuren in de registers bewaard. Dit geldt natuurlijk met name voor de programmeurs in assembler maar ook in 'C' heeft men de mogelijkheid door de keuze van zogenaamde register-variabelen het gebruik van registers te optimaliseren. Als we naar het instructieset en het programmeermodel van de 6502 kijken, dan lijken de locaties op pagina 0 heel veel op de registers van bijvoorbeeld een 68000.

Die mensen, die veel op de 6502 in assembler programmeren kennen ook het grote nadeel van het pagina 0 concept. Aangezien iedereen tracht zoveel mogelijk van de variabelen op pagina 0 onder te brengen, betekent dit vaak dat vrije ruimte op pagina 0 schaars is. Als er dan bovendien nog routines voor het afhandelen van interrupts zijn, dan wordt het helemaal dringen op pagina 0. Bij een processor met zo'n 16 registers is het gebruikelijk in zo'n geval enkele registers op de stack te schrijven maar bij 256 registers is dit ondoenlijk; bovendien is de stack van de 6502 ook slechts 256 byte groot.....

Een zeer fraaie oplossing zou het invoeren van een zogenaamd direct page register zijn. In dit register staat een adres en op dit adres begint pagina 0. Gaat men dan naar een ander deel van het programma, dan verandert men de inhoud van dit register en zie daar, men heeft weer 256 nieuwe pagina 0 bytes. Processoren met een dergelijke architectuur zijn inderdaad ontwikkeld (bijvoorbeeld de 65816).

Bij de 6502 wordt de snelheid van een computer voornamelijk bepaald door de snelheid waarmee gegevens van en naar het geheugen getransporteerd worden. Toen de technologie wat verder was en er chips gemaakt konden worden met zo'n 30.000 transistoren, heeft men onder andere hier iets aan proberen te doen. De snelheid van van de computer kan men op drie manieren verbeteren:

- Sneller geheugen. Was in de begintijd de toegangstijd tot het geheugen ongeveer 500 ns, momenteel is die tijd al teruggelopen tot ongeveer 100 ns of zelfs nog minder. Toch geldt nog steeds dat de snelheid van een systeem voor een belangrijk deel bepaald wordt door de snelheid van het geheugen.

- Meer informatie per memory-access transporteren. Men ging over van systemen die 8 bits per memory-access binnenhaalden naar systemen die in een keer 16 of zelfs 32 bits op kunnen halen of weg kunnen schrijven.

- Minder vaak het geheugen benaderen. Dit betekent dat men probeert meer informatie in de processor te bewaren hetgeen dus leidt naar meer registers of iets dergelijks.

Bij de 6502 en aanverwante systemen bleek bovendien de hoeveelheid geheugen dat geadresseerd kon worden een beperking. De programma's werden steeds groter en groter en het geheugen werd steeds goedkoper. (Tegenwoordig kost een megabyte aan geheugen ongeveer net zoveel als vroeger een kilobyte). Een van de methoden om dat op te lossen is het invoeren van een "PAGE"-register. Het geheugen is opgebouwd uit pagina's van een bepaalde grootte (bijvoorbeeld 64 kb). In het programma wordt dan met een twee-byte adres een locatie binnen deze pagina aangegeven. Door bij dit adres de inhoud van het page-register op te tellen, krijgen we het werkelijke absolute adres. Deze techniek doet heel veel denken aan de zojuist beschreven techniek van het direct page register. In wezen is het ook hetzelfde met dit verschil dat het direct page register uitsluitend betrekking heeft op de data terwijl het page register betrekking kan hebben op zowel code als data. De 65816 maakt dan ook gebruik van beide technieken terwijl men op de meer moderne versies van de PDP-11 nog een stap verder is gegaan. De programmeur heeft twee gescheiden adresruimten tot zijn beschikking. De eerste is alleen bedoeld voor data, de tweede is in principe uitsluitend voor programmaregels (code). Deze twee adresruimten hebben beide een eigen page-register.

Algemeen

Met zo´n 30.000 transistoren kunnen we een processor maken zoals bijvoorbeeld de 8086. Deze processor heeft 16 bits registers, een 16 bits databus en benadert het geheugen middels een page-register.

Een groot nadeel van het adresseren van het geheugen via een page-register is het feit dat programma-segmenten en datastructuren nooit groter mogen zijn dan 64 kb. De auteur heeft ooit eens een keer een groot systeem geprgrammeerd op een PDP-11/24. Deze machine werkt ook met een page-register waarbij de beperking zodanig is dat de geheugenruimte voor het programma + de data nooit groter mag worden dan 64 kb (voor de insiders: op de 11/24 is een scheiding tussen I- en D- ruimte niet mogelijk). Bovendien was van deze 64 kb al de helft gereserveerd voor zgn. common datastructuren. Dit betekende dat toch vrij omvangrijke applicatie-programma´s in 32 kb geprogrameerd moesten worden (in pascal). Een mogelijke oplossing voor dergelijke problemen is het werken met overlays. Dit houdt in dat het programma opgedeeld wordt in een aantal stukken die nooit gelijktijdig door de processor benaderd hoeven te worden. Voor elk stuk is dan, samen met de data, de genoemde 64 kb beschikbaar. Nou, deze techniek heeft ons toen heel wat hoofdbrekens gekost. In één geval bestond het programma uit 23 ! verschillende overlays.

Na verloop van tijd, werd het mogelijk nog meer transistoren op een chip te krijgen. De chip van de 68000 bevat bijvoorbeeld zo´n 70.000 transistoren. Wat heeft men daar zoal mee gedaan? In de eerste plaats heeft men een zestiental registers gedefinieerd van elk 32 bits. Deze registers zijn opgedeeld in 8 adres- en 8 dataregisters. De dataregisters zijn uiteraard bestemd voor data, de adresregisters kunnen goed gebruikt worden voor het doen van adresberekeningen. Het gedoe met page-registers is ook niet meer nodig, de 68000 kan direct 32 adresbits aansturen hetgeen overeenkomt met iets van twee gigabyte. Weliswaar zijn op de 68000 slechts 24 adresbits daadwerkelijk naar buiten gevoerd (nog altijd goed voor 16 Mb) doch bij de 68012 kan men al gebruik maken van het hele geheugenbereik. Verder heeft men het aantal instructies flink opgevoerd. Er zijn instructies voor vermenigvuldigen en delen, diverse andere arithmetische en logische instructies, een aantal move-instructies en zelfs een for-loop. Ook het aantal adresseermogelijkheden is groter dan bij de 6502. Bovendien kan men bij de meeste instructies aangeven of men een operand van 8, 16 of zelfs 32 bits wil benaderen.

Een summiere beschrijving van de 68000 staat in mijn artikel in de 6502 Kenner nr. 45 [2].

Natuurlijk zijn er aan het ontwerp van de 68000 ook nadelen verbonden. In de eerste plaats is daar het feit van het aantal instructies. Om het grote aantal verschillende instructies, met alle mogelijke combinaties van registers, adresseermogelijkheden en lengten van operanden te kunnen coderen, heeft de 68000 16 bits nodig. Bij de meeste instructies komen daar nog twee byte bij om een operand te bereiken; dit kunnen er echter ook vier zijn. Een instructie bestaat dus uit minimaal 2 byte en maximaal 6 byte. (Bij de 6502 is dit de helft !).

Een tweede nadeel zien we bij een interrupt. Omdat de 68000 16 registers heeft, moet men bij een interrupt en vaak ook bij een subroutine 15 registers op de stack redden (de zestiende is de stackpointer; die niet gered hoeft te worden). Dit zijn in totaal 60 byte en dus bij een 16 bits databus 30 schrijfoperaties aan het begin van de routine en 30 leesoperaties aan het einde. Bij een 6502 heeft men slechts 3 registers plus een stackpointer en kan men dit klusje in drie schrijf- en drie leesoperaties klaren.

Een derde nadeel ligt iets minder voor de hand. Omdat men een zeer groot aantal instructies heeft, heeft men in de processor weer een processor gebouwd met daarbij voor elke instructie een stukje programma. Dit programma heet microcode. Omdat men veel instructies heeft, heeft men dus ook veel microcode nodig en bovendien bij gecompliceerde instructies veel tijd om deze code uit te voeren. Bij de 6502 duurt de langste instructie 7 klokpulsen, een gemiddelde instructie 3 of 4 klokpulsen. Bij de 68000 kan het aantal klokpulsen oplopen tot meer dan 100. Een gemiddelde instructie duurt zo´n 10 klokpulsen.

De technologie heeft zich verder ontwikkeld. Momenteel kunnen er op een chip structuren gemaakt worden met afmetingen kleiner dan een micron. Op deze manier kan men nu iets van 200.000 transistoren op een chip kwijt. De vraag is nu wat we met al deze transistoren gaan doen.

CISC hoe ingewikkelder hoe beter.

De ene weg waarlangs men bezig is is de ontwikkeling van nog gecompliceerdere processoren. Momenteel worden de 80386 en de

68020 in veel systemen toegepast. Het zijn beide processoren met een volledige 32 bits adres- en databus, 32 bits registers en zeer gecompiceerde instructiesets. De 68030 is als opvolger van de 68020 reeds verkrijgbaar. Deze processor heeft onder andere een geïntergreerde memory management unit waarmee men op eenvoudige wijze een multitasking operating systeem zoals UNIX kan implementeren. Verder heeft Motorola de 68040 al aangekondigd.

Voor- en nadelen van CISC-architecturen

Zoals bij de bespreking van de 68000 al opgemerkt is, heeft een dergelijk ontwerp enkele meer of minder grote nadelen. Laten we de voor- en nadelen van de diverse ontwerpen eens op een rijtje zetten en proberen daar een afweging van te maken.

De 6502 heeft als voordeel de eenvoudige opbouw van zijn instructieset. Met behulp van 8 bits kan men het volledige instructieset inclusief alle adresseermogelijkheden coderen.

Een tweede voordeel is het feit dat de diverse instructies weinig klokcycli nodig hebben voor hun uitvoering.

De stap naar processors zoals de 8086 bracht een 16 bits databus en meer en bredere registers.

Het grote voordeel van de 68000 is zijn grote aantal registers met een breedte van 32 bits. Deze registers hebben echter als nadeel dat ze geregeld in het geheugen gered moeten worden hetgeen veel extra heen en weer getransporteer van data inhoudt. Verder heeft de 68000 zijn grote, lineaire geheugen als voordeel.

Kijken we naar de processoren van de categorie 68020 dan zien we daar een scheiding van adresruimten voor code en data naar voren komen. Eén van de pioniers op computergebied, Von Neumann, heeft in zijn machine het stored programm ontwerp uitgevonden. Dit hield in dat het programma in het geheugen, net zo als de data, opgeslagen werd. Dit betekende bovendien dat men aan de inhoud van een bepaald deel van het geheugen niet kon zien of het code of data was. Slechts de inhoud van de program counter bepaalde of de inhoud van een locatie als code gebruikt werd. In dit concept was het dan ook zeer eenvoudig bepaalde geheugengebieden afwisselend als code en als data op te vatten. Het begrip "Self Modifying Code" is toen ontstaan. In de tijd van Von Neumann was het volstrekt logisch een machine op die manier te ontwerpen. Tengenwoordig wordt wel gesproken van de "Von Neumann Bottleneck". Alles (code en data) moet namelijk over één en dezelfde databus. Bovendien treedt er nog een ander probleem op. Er zijn tegenwoordig technieken om het geheugen veel sneller als normaal te benaderen, mits.... we meerdere op éénvolgende plaatsen achter elkaar benaderen. Aangezien tegewoordig code- en datagebieden volstrekt gesheiden zijn, benaderen we in de praktijk afwisselend geheugen in het codegebied en in het datagebied. Dit pleit dus voor een scheiding tussen deze twee gebieden. Er zijn momenteel zelfs al processoren aangekondigd die hierom twee adres- en twee databussen hebben (Harvard Architectuur, in tegenstelling tot de Von Neumann Architectuur).

Waarom zijn tegenwoordig code- en datagebieden volstrekt gescheiden? In de eerste plaats heeft dit te maken met het, moderne, gestructureerd programmeren. Men legt hierbij in het begin van de module (subroutine of programmasegment) vast welke data men wil gebruiken waarna de code van de module volgt. Een taal als Pascal is hiervan een duidelijk voorbeeld, in tegenstelling tot bijvoorbeeld Basic waarbij data midden in het programma ontstaat. Een tweede reden heeft te maken met multitasking en multiuser. Als er meerdere gebruikers (users) tegelijk gebruik maken van dezelfde computer kunnen ze toch allemaal hetzelfde programma (task) gebruiken, bijvoorbeeld de editor. Om een programma te kunnen gebruiken, moet hij geladen zijn in het geheugen. Als er tien users zijn die allen de editor gebruiken, hoeft het programma zelf toch slechts een keer aanwezig zijn. Wel moet iedere user zijn eigen data hebben. Door nu in het programma de code en de data te scheiden, kan men de code één maal in het geheugen hebben terwijl iedere user zijn eigen datagebied heeft.

Machines zoals de 6502, de 8086 en de 68000 hebben als nadeel dat hun snelheid afhankelijk is van de snelheid van het geheugen. Bij de 68020 en de 68030 is dit voor een deel opgelost door op de chip een beperkt, snel geheugen te maken. In dit zogenaamde cache-geheugen kan code en data gedurende langere tijd bewaard blijven zodat ze snel gelezen kan worden. Gaat men echter in dit gebied schrijven, dan moet de wijziging toch in het geheugen aangebracht worden.

De 8086-achtige machines hebben als groot nadeel hun page-register. Grote(re) datastructuren en programmasegmenten kunnen

maar moeilijk gehanteerd worden.

Bij de 68000 lijkt de omvang van het instructieset een voordeel. Is dat eigenlijk wel zo? In de eerste plaats kost het decoderen van een instructie relatief veel tijd. In de tweede plaats kost de uitvoering van de meer gecompliceerde instructies zeer veel tijd. En in de derde plaats, hoe vaak gebruiken we eigenlijk een unsigned divide instructie?

RISC: Back to nature ?

Op basis van het voorgaande kunnen we ook eens een processor trachten te ontwerpen.

We nemen een processor en zorgen er voor dat deze een beperkt aantal eenvoudige instructies heeft die dan wel allemaal in zeer weinig klokcycli uit te voeren zijn. In de onderstaande tabel staan de instructies van de ARM, het hard van de ARCHIMEDUS. (Bron [3]).

==========

1 - De registerinstructies

ADD	Optellen
ADC	Optellen met carry
SUB	Aftrekken
RSB	Omgekeerd aftrekken
SBC	Aftrekken met carry
RSC	Omgekeerd aftrekken met carry
AND	Logische and van twee registers
EOR	Logische exclusive or
ORR	Logische or
BIC	Bepaalde bits nul maken
CMP	Vergelijk
CMN	Vergelijk negatief
TST	Test of beide 1 zijn
TEQ	Test of beide gelijk zijn
MOV	Verplaats
MVN	Verplaats negatief
MUL	Vermenigvuldig
MLA	Vermenigvuldig met optelling

2 - de geheugeninstructies

STR	Store register
LDR	Load register
STM	Store meerdere registers
LDM	Load meerdere registers

3 - de spronginstructies

B	Spring
BL	Spring en onthoud waar je vandaan komt
SWI	Software interrupt

==========

Verder nemen we ook het accumulator concept van de 6502 over. Elke instructie uit de eerste groep werkt alleen op registers. Bij de ARM kent een ADD-instructie drie operanden: de twee registers die bij elkaar opgeteld moeten worden en een derde register waar het resultaat naar toe moet. Het benaderen van het geheugen kan alleen via de LOAD en STORE instructies. Verder heeft de ARM nog een tweetal slimme toevoegingen:

- Bij een instructie kan men aangeven onder welke voorwaarde de instructie uitgevoerd moet worden.
- Men kan opgeven dat de tweede operand van de bewerking eerst nog een schuifoperatie moet ondergaan.

Uit [3] komt het volgende voorbeeld:

Het Basic statement

```
IF A<5000 THEN A=A*5
```

zou er op een ARM als volgt uit kunnen zien:

```
CMP r1,5000     vergelijk register 1
                met 5000
ADDCC r1,r1,r1  LSL 2
                als aan de voorgaande
                vergelijking voldaan
                is, tel dan r1 bij
                vier maal r1 op; het
                resultaat komt in r1.
```

Zowel de vergelijk- als de optelinstructie worden op de ARM in één klokcyclus uitgevoerd. Dit geldt trouwens voor alle instructies uit groep 1!

De snelheid waarmee de instructies uitgevoerd kunnen worden wordt veroorzaakt door het feit dat de instructies niet in microcode maar in echte transistoren etc. uitgevoerd zijn (dit kan bij weinig instructies). Bovendien heeft de ARM een vorm van pipelineing; de eerste instructie wordt uitgevoerd terwijl de tweede gedecodeerd en de derde uit het geheugen ingelezen wordt. Kenmerk van alle RISC architecturen is de eenvoud van de instructieset. De processor bezit slechts weinig instructies maar deze zijn wel zo snel en krachtig dat men met dit kleine aantal instructies op eenvoudige wijze subroutines voor gecompliceerde opgaven kan maken.

Van de 68020 nemen we de breedte van de databus en de registers over: 32 bits. Gaan we echt veelvuldig met floating point getallen aan de slag, dan valt 64 bits ook

te overwegen. (De ARM heeft een 32 bits databus). Ook de adresbus moet van voldoende omvang zijn (bij de ARM 26 bits).

En tenslotte de registers. Veel registers zijn handig omdat we dan weinig LOAD en STORE operaties van en naar het voor de processor zo trage geheugen hoeven uit te voeren. Weinig registers heeft als voordeel dat we bij een subroutine of interrupt weinig hoeven te redden. Op dit punt heeft men bij een aantal RISC-processoren (o.a. de SPARC van SUN) iets zeer slims bedacht. Voor een subroutine zijn een aantal (bijvoorbeeld 32) registers beschikbaar. Een deel van de registers (in ons voorbeeld 10) zijn globaal en dus overal, in elke subroutine te gebruiken. Deze zijn dus te vergelijken met de globale variabelen in een Pascal-programma. Een tweede deel (in ons voorbeeld 6) dient voor het doorgeven van parameters tussen de subroutines. In de meeste meeste gevallen wordt daarvoor de stack gebruikt hetgeen weer memory access betekent. De belangrijke parameters kunnen in dit concept dus per register doorgegeven worden. De rest van de registers is zuiver lokaal. Dit wil zeggen dat als we naar een ander niveau gaan (subroutine call of return) we een nieuw set registers krijgen. De processor heeft voldoende registers (in ons voorbeeld in totaal 132) om meerdere niveaus gelijktijdig in registers op te slaan, dus zonder het tijdrovende redden en terughalen uit geheugen. In het genoemde voorbeeld kan men tot 8 niveaus opslaan. Gaat men nog dieper, hetgeen eigenlijk niet voorkomt, dan worden er weer registers in het geheugen gered. Het voorbeeld is beschreven in [4].

Door op de hierboven beschreven manier een processor te ontwerpen, heeft men de twee meest wezenlijke zaken ondervangen. Men heeft in de eerste plaats de uitvoeringstijd van de instructies drastisch weten te beperken. Verder heeft men niet zozeer de snelheid waarmee het geheugen benaderd kan worden vergroot als wel het aantal keren dat het geheugen benaderd wordt flink verkleind. Op deze manier is er een processor ontstaan die zeer snel is. Dat een dergelijk ontwerp echt kan werken blijkt uit de volgende resultaten.

Met de BASIC-benchmarks van het engelse blad "Personal Computer World" zijn als resultaten verkregen:

Apple Macintosh:	12.1 s
IBM-PC:	17.6 s
IBM-PC/AT:	7.1 s
ARM	0.7 s

Daar de ARM zeker niet het meest geavanceerde op RISC-gebied is, is nu al wel duidelijk dat de RISC-architecturen zeer veel in zich hebben en waarschijnlijk zeer geduchte concurrenten voor de CISC-architecturen zullen worden.

Ook Motorola, als fabrikant van zeer gecompliceerde processoren, heeft een RISC architectuur aangekondigd. Deze processor zal waarschijnlijk begin 1989 beschikbaar komen. Het is de 88000 die tesamen met de memory-management chips 88100 en 88200 de processor gaan vormen. Deze processor heeft eigenlijk alles in zich wat in het voorgaande beschreven is; Een RISC instructieset, een registerfile de Harvard architectuur met compleet gescheiden adres- en databussen. Bovendien heeft de 88000 ook nog een floating-point unit die op dezelfde chip als de processor gebouwd is doch volledig onafhankelijk, als coprocessor zijn werk doet. De beschrijving van de 88000 en van de Am29000 van AMD kan men o.a. vinden in [5]. Bovendien bevat dit artikel interssante gegevens over moderne benaderingstechnieken van geheugens.

Literatuur

Dit artikel is gebaseerd op de volgende publicaties:

1: Pete Wilson: The CPU WARS (Byte may 1988 pag. 213).

2: Gert van Opbroek: The MC68000; a new processor in our club (De 6502 Kenner nr. 45 Augustus 1986 pag. 7).

3: Jacques Haubrich: De ARM (HCC-nieuwsbrief nr. 102 maart 1988 pag. 59).

4: Guido Treutwein: RISC-Chips: Weniger Befehle - mehr Leistung (MC 2, februari 1986 pag. 90).

5: Trevor Marshall: Real-World RISCs (Byte may 1988 pag. 263).

==========

```
OS65D EXTENSIONS              PROTON 650X ASSEMBLER  V4.4   PAGE: 0001

0001  0000                   .TIT 'OS65D EXTENSIONS'
0002  0000                   .OPT SYMBOL,NOM
0003  0000                   .LOCAL
0004  0000            ; ************************************
0005  0000            ; *       DOS ERRORS IN WORDS        *
0006  0000            ; ************************************
0007  0000            ;
0008  0000            ; In de originele versie van OS65D worden foutmel-
0009  0000            ; dingen via een cijfer/letter van 1 t/m D aangegeven
0010  0000            ; en bij de Elektuur uitbreiding ook met #E en #F.
0011  0000            ; Deze routine heeft tot doel de foutmelingen van
0012  0000            ; DOS (inclusief E en F) in teksten af te drukken.
0013  0000            ; Voor de uitbreiding is een niet meer gebruikt geheu-
0014  0000            ; gengedeelte van de DOS benut.
0015  0000            ;
0016  0000            ; DATE: MEI 1988, BERT VAN TIEL.
0017  0000            ;
0018  0000            ; Noot van de redactie:
0019  0000            ;    Om formaat-technische redenen, is de oorspronkelijke
0020  0000            ;    versie, in Micro-Ade formaat, geconverteerd naar
0021  0000            ;    het formaat van de Proton-assembler, ook bekend onder
0022  0000            ;    de naam (Octo-)Fate.
0023  0000            ;
0024  0000            ; WIJZIGINGEN IN DOS
0025  0000            ; ------------------
0026  0000            ;
0027  0000                         * = $2A4B
0028  2A4B  20CA2A               JSR DISKER
0029  2A4E            ;
0030  2A4E            ; * USED ADDRESSES AND ROUTINES *
0031  2A4E            ;
0032  2A4E           PRCHA     = $2343
0033  2A4E           UNLDHD    = $2761
0034  2A4E            ;
0035  2A4E                         * = $2AC4
0036  2AC4            ;
0037  2AC4  00       OSERR     .BYT $00
0038  2AC5  00                 .BYT $00
0039  2AC6  01                 .BYT $01 DEFAULT I/O DEVICES
0040  2AC7  00                 .BYT $00
0041  2AC8  00                 .BYT $00
0042  2AC9  00                 .BYT $00
0043  2ACA            ;
0044  2ACA  AA       DISKER    TAX              ; A BEVAT HET FOUTNUMMER
0045  2ACB  C90E               CMP #$0E         ; FOUTNUMMER >= E?
0046  2ACD  3003               BMI ERRTMD
0047  2ACF  4C4F34             JMP ERREF
0048  2AD2  CA       ERRTMD    DEX              ; GEBRUIK DIT OM DE Y-INDEX IN
0049  2AD3  BC5033             LDY TABLEA,X     ; DE ASCII TABEL TE VINDEN
```

OS65D EXTENSIONS PROTON 650X ASSEMBLER V4.4 PAGE: 0002

```
0050  2AD6  B95D33 PRINTE   LDA TABLEB,Y      ; PRINT DE FOUTMELDING
0051  2AD9  F007            BEQ EPRINT
0052  2ADB  204323          JSR PRCHA
0053  2ADE  C8              INY
0054  2ADF  4CD62A          JMP PRINTE
0055  2AE2  4C6127 EPRINT   JMP UNLDHD        ; KOP OMHOOG EN EXIT
0056  2AE5           ;
0057  2AE5           ; UITBREIDING
0058  2AE5           ; -----------
0059  2AE5           ;
0060  2AE5                  * = $3350
0061  3350           ;
0062  3350  00      TABLEA   .BYT $00          ; ERR1
0063  3351  0D              .BYT $0D          ; ERR2
0064  3352  1A              .BYT $1A          ; ERR3
0065  3353  2E              .BYT $2E          ; ERR4
0066  3354  46              .BYT $46          ; ERR5
0067  3355  51              .BYT $51          ; ERR6
0068  3356  61              .BYT $61          ; ERR7
0069  3357  6E              .BYT $6E          ; ERR8
0070  3358  7F              .BYT $7F          ; ERR9
0071  3359  97              .BYT $97          ; ERRA
0072  335A  B0              .BYT $B0          ; ERRB
0073  335B  C2              .BYT $C2          ; ERRC
0074  335C  D5              .BYT $D5          ; ERRD
0075  335D           ;
0076  335D  5041    TABLEB   .BYT 'PARITY ERROR' ,$00
0077  336A  5245             .BYT 'REREAD ERROR' ,$00
0078  3377  43414E           .BYT 'CAN' ,$27,'T WRITE TRACK 0',$00
0079  338B  4449             .BYT 'DISK IS WRITE PROTECTED' ,$00
0080  33A3  5345             .BYT 'SEEK ERROR' ,$00
0081  33AE  4452             .BYT 'DRIVE NOT READY' ,$00
0082  33BE  5359             .BYT 'SYNTAX ERROR' ,$00
0083  33CB  4241             .BYT 'BAD TRACK NUMBER' ,$00
0084  33DC  43414E           .BYT 'CAN' ,$27,'T FIND TRACK HEADER',$00
0085  33F4  43414E           .BYT 'CAN' ,$27,'T FIND SECTOR HEADER',$00
0086  340D  4241             .BYT 'BAD SECTOR LENGTH' ,$00
0087  341F  4649             .BYT 'FILE DOESN' ,$27,'T EXIST',$00
0088  3432  522F             .BYT 'R/W ATTEMPT PAST END OF FILE' ,$00
0089  344F           ;
0090  344F  8A      ERREF    TXA
0091  3450  E90E             SBC #$0E
0092  3452  AA              TAX
0093  3453  BC6534           LDY TABLEC,X
0094  3456  B96734 PRINTF   LDA TABLED,Y
0095  3459  F007            BEQ FPRINT
0096  345B  204323          JSR PRCHA
0097  345E  C8              INY
0098  345F  4C5634          JMP PRINTF
0099  3462  4C6127 FPRINT   JMP UNLDHD
0100  3465           ;
0101  3465  00      TABLEC   .BYT $00          ; ERRE
```

```
OS65D EXTENSIONS              PROTON 650X ASSEMBLER  V4.4   PAGE: 0003


0102  3466  1C              .BYT $1C          ; ERRF
0103  3467                ;
0104  3467  4E4F    TABLED  .BYT ´NOT ENOUGH TRACKS AVAILABLE´ ,$00
0105  3483  4449            .BYT ´DIRECTORY FULL´ ,$00
0106  3492                ;
0107  3492                  .LOCAL
0108  3492                ;
0109  3492                ; ************************************
0110  3492                ; *       === DIR === EXTENSION 2    *
0111  3492                ; ************************************
0112  3492                ;
0113  3492                ; Dit is een uibreiding op de bestaande DI opdracht
0114  3492                ; van DOS V3.3 uitgebreid door Elektuur ($E5E1-E653).
0115  3492                ; Nu zal ook de 2e helft van de directory
0116  3492                ; worden getoond, EN WEL IN 4 KOLOMMEN NAAST ELKAAR.
0117  3492                ;
0118  3492                ; DATE: MEI 1988, BERT VAN TIEL.
0119  3492                ;
0120  3492                ; Noot van de redactie:
0121  3492                ;    Om formaat-technische redenen, is de oorspronkelijke
0122  3492                ;    versie, in Micro-Ade formaat, geconverteerd naar
0123  3492                ;    het formaat van de Proton-assembler, ook bekend onder
0124  3492                ;    de naam (Octo-)Fate.
0125  3492                ;
0126  3492                ; WIJZIGINGEN IN DOS
0127  3492                ; ------------------
0128  3492                ;
0129  3492                      * = $E5E1
0130  E5E1                ;
0131  E5E1                ; * USED ADDRESSES/ROUTINES *
0132  E5E1                ;
0133  E5E1              PTR      = $0010
0134  E5E1              OSIBAD   = $00E1          ; POINTER TO DOS BUFFER
0135  E5E1              PRINT    = $2343          ; PRINT CHAR. IN ACCU
0136  E5E1              SECTNM   = $265E
0137  E5E1              DIR      = $2B29          ; DI UNDER V3.3 (DIR/TRACK)
0138  E5E1              BULEN    = $2CED          ; CURRENT BUFFER LENGTH
0139  E5E1              CRLF     = $2D6A
0140  E5E1              STROUT   = $2D73          ; STRING OUTPUT
0141  E5E1              PRTAHX   = $2D92          ; PRINT 2 CHAR. OF ACCU
0142  E5E1              RDDIR    = $E5C5          ; READ DIRECTORY FROM DISK
0143  E5E1                ;
0144  E5E1  4C292B NDIRY    JMP DIR
0145  E5E4  A002   NEWDIR   LDY #$02
0146  E5E6  B1E1           LDA (OSIBAD),Y
0147  E5E8  C90D           CMP #$0D          ; CHECK FOR DI CR
0148  E5EA  F007           BEQ NDIRX
0149  E5EC  ADED2C         LDA BULEN
0150  E5EF  C902           CMP #$02
0151  E5F1  D0EE           BNE NDIRY         ; CHECK FOR DI UNDER V3.3
0152  E5F3  20732D NDIRX    JSR STROUT
0153  E5F6  0D             .BYT $0D,$0A,´==DIR==´,$00
```

```
OS65D EXTENSIONS              PROTON 650X ASSEMBLER  V4.4   PAGE: 0004


0154  E600  206A2D          JSR CRLF
0155  E603  A201            LDX #$01
0156  E605  20C5E5 NDIRG    JSR RDDIR          ; READ DIRECTORY FROM DISK
0157  E608  A200            LDX #$00
0158  E60A  A000   NDIRA    LDY #$00
0159  E60C  4C9234          JMP NDIRB
0160  E60F  A000   NDIRBB   LDY #$00
0161  E611  B110   NDIRD    LDA (PTR),Y
0162  E613  204323          JSR PRINT          ; OUTPUT FILE NAME
0163  E616  C8              INY
0164  E617  C006            CPY #$06
0165  E619  D0F6            BNE NDIRD
0166  E61B  A920            LDA #' '
0167  E61D  204323          JSR PRINT
0168  E620  B110            LDA (PTR),Y
0169  E622  20922D          JSR PRTAHX         ; OUTPUT TRACK RANGE
0170  E625  20732D          JSR STROUT
0171  E628  202D20          .BYT ' - ' ,$00
0172  E62C            ;
0173  E62C  A007            LDY #$07
0174  E62E  B110            LDA (PTR),Y
0175  E630  20922D          JSR PRTAHX
0176  E633  20732D          JSR STROUT
0177  E636  2020            .BYT '   ' ,$00
0178  E63B            ;
0179  E63B  18      NDIRE   CLC
0180  E63C  A510            LDA PTR
0181  E63E  6908            ADC #$08           ; POINT TO NEXT ENTRY IN DIRECTORY
0182  E640  8510            STA PTR
0183  E642  9002            BCC NDIRF
0184  E644  E611            INC PTR+01
0185  E646            ;
0186  E646  C979    NDIRF   CMP #$79
0187  E648  D0C0            BNE NDIRA          ; CHECK FOR END OF DIRECTORY
0188  E64A  A511            LDA PTR+01
0189  E64C  C92F            CMP #$2F
0190  E64E  D0BA            BNE NDIRA
0191  E650  20AD34          JSR DIREXT
0192  E653  60              RTS
0193  E654            ;
0194  E654            ; UITBREIDING
0195  E654            ; -----------
0196  E654            ;
0197  E654                  * = $3492
0198  3492            ;
0199  3492  B110    NDIRB   LDA (PTR),Y
0200  3494  C923            CMP #'#'
0201  3496  D008            BNE NDIRC          ; CHECK FOR EMPTY ENTRY
0202  3498  C8              INY
0203  3499  C006            CPY #$06
0204  349B  D0F5            BNE NDIRB
0205  349D  4C3BE6          JMP NDIRE
```

```
OS65D EXTENSIONS                PROTON 650X ASSEMBLER  V4.4   PAGE: 0005

0206  34A0            ;
0207  34A0  E8      NDIRC    INX
0208  34A1  E005             CPX #$05
0209  34A3  D005             BNE RET
0210  34A5  A201             LDX #$01
0211  34A7  206A2D           JSR CRLF
0212  34AA  4C0FE6 RET        JMP NDIRBB
0213  34AD            ;
0214  34AD  206A2D DIREXT    JSR CRLF
0215  34B0  A202             LDX #$02          ; X IS SECTOR 2 (TRACK 12)
0216  34B2  EC5E26           CPX SECTNM        ; SECTOR 2 AL GEHAD?
0217  34B5  D003             BNE DEA           ; NEE
0218  34B7  4C6A2D           JMP CRLF          ; MAAK BESTAANDE ROUTINE AF
0219  34BA  4C05E6 DEA        JMP NDIRG         ; TOON 2E HELFT DIRECTORY
0220  34BD            ;
0221  34BD                    .LOCAL
0222  34BD            ;
0223  34BD            ; ***************************************
0224  34BD            ; *       === PUT === EXTENSION 2       *
0225  34BD            ; ***************************************
0226  34BD            ;
0227  34BD            ; DIT IS EEN AANVULLING OP DE BESTAANDE DOSPUT EXT.
0228  34BD            ; VAN ELEKTUUR. ($E47C-E5AD).
0229  34BD            ; HET IS HIERMEE MOGELIJK OM IN DE TWEEDE HELFT VAN
0230  34BD            ; DE DIRECTORY ENTRY´S TE MAKEN, DUS MAX. 64 I.P.V. 32.
0231  34BD            ; HIERVOOR ZIJN OOK "DE FOUTMELINGEN IN WOORDEN"
0232  34BD            ; AANGEVULD. (ZIE DOS ERROS IN WORDS).
0233  34BD            ;
0234  34BD            ; DATE: MEI 1988, BERT VAN TIEL.
0235  34BD            ;
0236  34BD            ; Noot van de redactie:
0237  34BD            ;    Om formaat-technische redenen, is de oorspronkelijke
0238  34BD            ;    versie, in Micro-Ade formaat, geconverteerd naar
0239  34BD            ;    het formaat van de Proton-assembler, ook bekend onder
0240  34BD            ;    de naam (Octo-)Fate.
0241  34BD            ;
0242  34BD            ; WIJZIGINGEN IN DOS EXT. ELEKTUUR
0243  34BD            ; ------------------------------
0244  34BD            ;
0245  34BD                    * = $E52F
0246  E52F  4CBD34           JMP PUEXT
0247  E532                    * = $E5A4
0248  E5A4  EA                NOP
0249  E5A5  EA                NOP
0250  E5A6  EA                NOP
0251  E5A7  EA                NOP
0252  E5A8  EA                NOP
0253  E5A9            ;
0254  E5A9            ; * USED ADDRESSES/ROUTINES *
0255  E5A9            ;
0256  E5A9          SECTNM   = $265E           ; SECTOR NUMBER
0257  E5A9          PUEXE    = $E4A7
```

```
OS65D EXTENSIONS                PROTON 650X ASSEMBLER  V4.4   PAGE: 0006


0258  E5A9           ERRENT   = $2A4B
0259  E5A9            ;
0260  E5A9                    * = $34BD
0261  34BD            ;
0262  34BD  A202     PUEXT    LDX #$02
0263  34BF  EC5E26             CPX SECTNM
0264  34C2  F003               BEQ ERRF
0265  34C4  4CA7E4             JMP PUEXE
0266  34C7  A90F     ERRF     LDA #$0F          ; ERROR #F DIRECTORY FULL
0267  34C9  4C4B2A             JMP ERRENT
0268  34CC                     .END

SYMBOL     VALUE
PRCHA  02  2343     UNLDHD 02  2761     OSERR  02  2AC4
DISKER 02  2ACA     ERRTMD 02  2AD2     PRINTE 02  2AD6
EPRINT 02  2AE2     TABLEA 02  3350     TABLEB 02  335D
ERREF  02  344F     PRINTF 02  3456     FPRINT 02  3462
TABLEC 02  3465     TABLED 02  3467     PTR    03  0010
OSIBAD 03  00E1     PRINT  03  2343     SECTNM 03  265E
DIR    03  2B29     BULEN  03  2CED     CRLF   03  2D6A
STROUT 03  2D73     PRTAHX 03  2D92     RDDIR  03  E5C5
NDIRY  03  E5E1     NEWDIR 03  E5E4     NDIRX  03  E5F3
NDIRG  03  E605     NDIRA  03  E60A     NDIRBB 03  E60F
NDIRD  03  E611     NDIRE  03  E63B     NDIRF  03  E646
NDIRB  03  3492     NDIRC  03  34A0     RET    03  34AA
DIREXT 03  34AD     DEA    03  34BA     SECTNM 04  265E
PUEXE  04  E4A7     ERRENT 04  2A4B     PUEXT  04  34BD
ERRF   04  34C7
```

Binnenkort in de 6502 Kenner

In het kader van de wijziging van de doelstelling van de vereniging willen we in de komende jaargang onder andere over de volgende onderwerpen artikelen gaan schrijven:

- Single Chip processors. Dit artikel zal gepubliceerd worden naar aanleiding van de lezing van Adri Hankel in september 1988 in Haarlem.
- De kwaliteitszorg van software. Dit artikel zal door Ruud Uphoff geschreven worden als samenvatting van zijn lezing op de bijeenkomst van mei 1988 in Krimpen a.d. IJssel.
- Programmeerbare hardware. Tegenwoordig zijn er niet alleen Eproms en EEproms maar ook PALlen en GALlen. Aan Nico de Vries is gevraagd hierover een publicatie te schrijven.
- Grafische Pocessoren. Gert van Opbroek is van plan hierover binnenkort een artikel te schrijven. Als aanvulling hierop, zou het zinvol zijn wat grafische software te plaatsen; wie heeft er nog iets of wil iets schrijven?
- Het formaat van floating point getallen. Ook hierover wil Gert van Opbroek een artikel schrijven. Als hij voldoende tijd heeft, zal dit bovendien gepaard gaan met een floating point pakket voor een Forthsysteem, draaiend op een 6502.
- De DOS-65 virtuele diskkaart. Deze kaart is al vaak aangekondigd maar bestaat nog steeds niet. Het probleem zit hem in het aantal IC´s dat op één europrint ondegebracht moet worden. Het lijkt er echter toch op dat in de komende winter deze kaart echt beschikbaar komt.

KERMIT, het communicatieprogramma.

Gert van Opbroek
Bateweg 60
2481 AN Woubrugge
01729-8636

Inleiding.

Dit artikel kan gezien worden als het vervolg op het artikel van Gert Klein in de 6502 Kenner nummer 54. In dat artikel wordt beschreven hoe het Kermit-**protocol** werkt. Daar het Kermit-protocol meestal binnen het Kermit programma gebruikt wordt, hierbij een artikeltje waarin dit programma beschreven wordt.

Elk Kermit communicatieprogramma ondersteunt voor de uitwisseling van files het Kermit protocol, het is echter niet zo, dat het Kermit protocol alleen in een Kermit communicatieprogramma voorkomt. Een goed voorbeeld hiervan is bijvoorbeeld OPUS, het programma dat op ons Bulletin Board draait en Procomm, het communicatieprogramma dat vooral in de MS-DOS wereld bekend is.

Kermit, zowel het protocol als het communicatiepakket is in eerste instantie ontwikkeld door Frank da Cruz en Bill Catchings aan de Columbia University te New York. Het programma is ontwikkeld om informatie uit te wisselen tussen een DEC 20, een IBM 370 en diverse microcomputers. De eerste versies werden geprogrammeerd op de DEC 20 en onder CPM/80. Al spoedig volgden versies voor IBM VM/CMS en PC-DOS. Dit alles vond plaats in 1981.

Een oorspronkelijke versie van Kermit stond in de Kermit Protocol Manual. Dit document beschrijft behalve het Kermit protocol ook een communicatieprogramma waarbinnen dit protocol werkzaam is. In de oudere versies van dit document was bovendien een C-source voor een Kermit-implementatie bijgevoegd. In de huidige (6e) editie is deze source niet meer aanwezig vanwege o.a. het feit dat het niet een goed voorbeeld voor eventuele andere programmeurs was. Naar aanleiding van de Kermit Protocol Manual zijn er enkele honderden Kermit implementaties gemaakt voor bijna even zoveel computers. Deze versies zijn niet alleen in ´C´ maar ook in andere talen en diverse assemblers geschreven. De Kermit voor DOS-65 is ontwikkeld vanuit de Apple Kermit en is geschreven in assembler. Kermit voor de VAX is geschreven in BLISS.

Zo rond 1985 is er een nieuwe implementatie van Kermit gemaakt, C-Kermit genaamd, speciaal ontwikkeld om onder UNIX te draaien. Deze versie is zo´n beetje de meest uitgebreide Kermit-implementatie en is in ´C´ geschreven. Ook deze versie is onderhand geporteerd naar vele machines (o.a. de op 68000 gebaseerde systemen met OS9/68k als operating systeem). In dit artikel wordt de C-Kermit versie 4E(068) van januari 1988 als voorbeeld gebruikt.

Alles wat met Kermit te maken heeft is vrij beschikbaar. Dit omvat sources, documentatie en executables. Bij de verspreiding van Kermit mag men alleen materiaalkosten in rekening brengen. Verder heeft Kermit een officiele beheerder. Dit is:

> Kermit Distribution
> Columbia University for Computing Activities
> 7th Floor, Watson Laboratory
> 612 West 115th Street
> New York, NY 10025

Dit instituut tracht de Kermit-implementaties voor de diverse machines te verzamelen en coördineert de release van nieuwe versies. Voor de verspreiding van Kermit wordt o.a. gebruik gemaakt van internationale computernetwerken (BITNET). Kermit is geen Public Domain, hoewel er in de praktijk weinig verschillen zijn. In de sources en manuals staan zgn. copyright notices die ervoor dienen dat niet op een kwade dag het werk van deze mensen op een comercieële basis door derden uitgebracht wordt. Een van de meest opvallende regels in de Kermit documentatie is verder de volgende:

--- PLEASE USE KERMIT ONLY FOR PEACEFUL AND HUMAINE PURPOSES ---

waar ik mij graag bij aansluit.

Mogelijkheden

Men tracht ook de Kermit documentatie te standaardiseren. Elke Kermit User Guide hoort te beginnen met een overzicht van de mogelijkheden. Hieronder staat dit overzicht van UNIX Kermit, overgenomen uit de originele documentatie (vandaar het engels).

Unix Kermit Capabilities At A Glance:

Localoperation	Yes
Remote operation	Yes
Login scripts	Yes (UUCP style)
Transfer text files	Yes
Transfer binary files	Yes
Wildcard send	Yes
File transfer interruption	Yes
Filename collision avoidance	Yes
Can time out	Yes
8th-bit prefixing	Yes
Repeat count prefixing	Yes
Alternate block checks	Yes
Terminal emulation	Yes
Communication settings	Yes
Transmit BREAK	Yes (most versions)
Support for dialout modems	Yes
IBM mainframe communication	Yes
Transaction logging	Yes
Session logging	Yes
Debug logging	Yes
Packet logging	Yes
Act as server	Yes
Talk to server	Yes
Advanced server commands	Yes
Local file management	Yes
Command/Init files	Yes
UUCP and multiuser line locking	Yes
Long packets	Yes
Sliding windows	No
File attributes packets	No
Raw file transmit	No

U ziet, een respectabele lijst van mogelijkheden. In de volgende paragrafen zal ik globaal beschrijven wat deze mogelijkheden inhouden en waarom ze zinvol zijn.

Local operation/Remote operation

Kermit is bedoeld voor file-transfer. Dit betekent dat twee Kermit´s elkaar de informatie toezenden. De Kermit die op het systeem van de gebruiker draait, meestal de micromputer, wordt de locale kermit genoemd. De Kermit aan de andere kant, bijvoorbeeld een mainframe, is de remote kermit. Een local-kermit geeft meestal status-informatie over de file transfer, de remote kermit vaak niet.

Het begrip remote Kermit is alleen van belang bij file-transfer. Daar Kermit echter een communicatiepakket is, kan men een local Kermit ook in de zogenaamde Connect mode bedrijven. Dit houdt in, dat het locale systeem een terminal vormt van het andere systeem. Men kan dan dus op het remote systeem werken alsof men aan een terminal op dit systeem werkt. Bij de bespreking van de terminal emulatie kom ik hier nog op terug.

Login scripts

Op grotere systemen en Bulletin Boards moet men zich meestal middels een kort vraag- en antwoordspel aan het systeem bekend maken,zoals bijvoorbeeld op ons Bulletin Board:

```
VOORNAAM:              gert
ACHTERNAAM:            van.opbroek
GERT VAN.OPBROEK ?     y
PASSWORD:              ...........
```

Bij een aantal Kermit-implementaties kan men een dergelijke dialoog inprogrammeren zodat het inloggen automatisch gaat.

Transfer text/binary files

Tekst- files kunnen alle Kermit-implementaties uitwisselen, daar zijn ze tenslotte voor gemaakt. Binaire files kunnen de meeste Kermits ook uitwisselen. Hiermee kan men bijvoorbeeld complete executables verzenden. Natuurlijk heeft het uitwisselen van executables alleen zin als de systemen hetzelfde zijn. Ook gecomprimeerde .ARC files zijn binair.

Voor het uitwisselen van files wordt door Kermit gebruik gemaakt van het door Gert Klein beschreven Kermit protocol. Dit protocol is zodanig van opzet, dat de twee Kermits aan het begin van de transfer afspreken welke van de hier besproken faciliteiten wel en welke niet gebruikt zullen worden. De bediener van de local Kermit kan hoogstens zijn voorkeur kenbaar maken doch als de remote kermit de gekozen faciliteit niet ondersteunt, dan wordt deze ook niet gebruikt.

Wildcard send

Dit houdt in dat men meerdere files die verzonden moeten worden aan kan geven door middel van zogenaamde wildcards. Een wildcard is een teken dat in de plaats staat voor een ander teken of andere tekens. Een paar voorbeelden:

A*.*	De ster betekent elk teken of reeks van tekens. In dit geval dus alle files die met A beginnen.
ABC??.PAS	Het vraagteken kan vervangen worden door elk teken doch door slechts een teken. In dit geval dus alle .PAS files van vijf letters, beginnend met ABC.

File transfer interruption

Men kan het overzenden van files onderbreken en op een nette manier afsluiten.

Filename collission avoidance

Onder Kermit wordt de filenaam meegezonden. In normale gevallen wordt de file onder dezelfde naam op het andere systeem opgeslagen voor zover de filenaam voldoet aan de conventies van het operating systeem. Men kan overigens aangeven dat de file een andere naam moet krijgen. Nu kan het dus gebeuren dat er een file overgestuurd wordt die dezelfde naam heeft als een andere file die al op het andere systeem aanwezig is. Kermit zorgt er dan voor dat de file een andere naam krijgt zodat de reeds aanwezige file niet overschreven wordt.

Can time out

Als er op een of andere manier iets fout gaat bij de filetransfer, is het handig als het systeem niet tot Sint Juttemis blijft wachten maar probeert de uitwisseling weer op gang te brengen en als dit niet gaat, stopt met een foutboodschap. Deze optie is in vrijwel alle Kermits aanwezig waarbij, bij de meer geavanceerde implementaties, de maximale wachttijd (time out) vaak instelbaar is.

8th-bit prefixing, Repeat Count prefixing

Zoals in het artikel van Gert Klein als is aangegeven, kunnen binaire files ook met het Kermit-protocol overgestuurd worden. Het protocol schrijft voor, dat alle stuurtekens door middel van een zogenaamde prefix aangegeven worden. cntrl-A (SOH) wordt aangegeven met #A. Ingeval van data waarbij het hoogste bit geset is, zijn er twee mogelijkheden: 1) Er wordt uitsluitend 7-bits data overgestuurd en 2) er wordt 8-bits data overgestuurd. Als er 7-bits data overgestuurd wordt, dan wordt er door middel van een prefix (in dit geval de &) aangegeven dat in het databyte bit 8 geset moet worden.

Repeat Count prefixing kan gebruikt worden om meerdere van dezelfde tekens achter elkaar op een korte manier aan te geven. Men kan op deze manier tot maximaal 94 van dezelfde tekens in 3 tekens aangeven. De onderstaande voorbeelden komen uit de Kermit Protocol Manual:

^ betekent Cntrl
´ betekent 8e bit geset
"(" is ASCII 40 - 32 = 8

Teken	Representatie met Prefix(es)	Met herhaling voor 8 tekens
A (41)	A	~(A
^A (01)	#A	~(#A
´A (C1)	&A	~(&A
´^A (81)	&#A	~(&#A

Kan men over de lijn 8 bit data sturen, dan zien de bovenstaande voorbeelden er als volgt uit:

Teken	Representatie met Prefix(es)	Met herhaling voor 8 tekens
A (41)	A	~(A
^A (01)	#A	~(#A
´A (C1)	´A	~(´A
´^A (81)	#´A	~(#´A

Noot ´ is dus geen prefix; dit betekent alleen dat er een byte met het 8e bit geset overgestuurd wordt.

Wordt er in de file een prefix-teken verstuurd, dan wordt dit teken vooraf gegaan door het # teken. Dus:

Teken:	Vestuurd:
#	##
&	#&
~	#~

ujl
Het verstuurde teken kan dan nog vooraf worden gegeaan door de 8th-bit prefix en de repeat count prefix.

Alternate Block Checks

Alle Kermit-implementaties ondersteunen op zijn minst de meest eenvoudige vorm van Block Check. Deze vorm is in het artikel van Gert Klein besproken. Een aantal Kermit-implementaties kennen verder nog twee andere vormen. De eerste is gebaseerd op een twaalf bits checksum; de tweede op de 16-bit CRC-CCITT die ook in de meeste netwerken gebruikt wordt. In het algemeen voldoet de eenvoudige vorm uitstekend. Heeft men echter binaire files die over slechte verbindingen verstuurd worden, dan is een alternatieve block check mogelijk iets beter, omdat eventuele fouten dan beter te detecteren zijn.

Terminal emulation

In een voorgaande paragraaf is al aangege-

Unix Kermit Capabilities At A Glance:

Localoperation	Yes
Remote operation	Yes
Login scripts	Yes (UUCP style)
Transfer text files	Yes
Transfer binary files	Yes
Wildcard send	Yes
File transfer interruption	Yes
Filename collision avoidance	Yes
Can time out	Yes
8th-bit prefixing	Yes
Repeat count prefixing	Yes
Alternate block checks	Yes
Terminal emulation	Yes
Communication settings	Yes
Transmit BREAK	Yes (most versions)
Support for dialout modems	Yes
IBM mainframe communication	Yes
Transaction logging	Yes
Session logging	Yes
Debug logging	Yes
Packet logging	Yes
Act as server	Yes
Talk to server	Yes
Advanced server commands	Yes
Local file management	Yes
Command/Init files	Yes
UUCP and multiuser line locking	Yes
Long packets	Yes
Sliding windows	No
File attributes packets	No
Raw file transmit	No

U ziet, een respectabele lijst van mogelijkheden. In de volgende paragrafen zal ik globaal beschrijven wat deze mogelijkheden inhouden en waarom ze zinvol zijn.

Local operation/Remote operation

Kermit is bedoeld voor file-transfer. Dit betekent dat twee Kermit´s elkaar de informatie toezenden. De Kermit die op het systeem van de gebruiker draait, meestal de micromputer, wordt de locale kermit genoemd. De Kermit aan de andere kant, bijvoorbeeld een mainframe, is de remote kermit. Een local-kermit geeft meestal status-informatie over de file transfer, de remote kermit vaak niet.

Het begrip remote Kermit is alleen van belang bij file-transfer. Daar Kermit echter een communicatiepakket is, kan men een local Kermit ook in de zogenaamde Connect mode bedrijven. Dit houdt in, dat het locale systeem een terminal vormt van het andere systeem. Men kan dan dus op het remote systeem werken alsof men aan een terminal op dit systeem werkt. Bij de bespreking van de terminal emulatie kom ik hier nog op terug.

Login scripts

Op grotere systemen en Bulletin Boards moet men zich meestal middels een kort vraag- en antwoordspel aan het systeem bekend maken,zoals bijvoorbeeld op ons Bulletin Board:

```
VOORNAAM:            gert
ACHTERNAAM:          van.opbroek
GERT VAN.OPBROEK ?   y
PASSWORD:            ............
```

Bij een aantal Kermit-implementaties kan men een dergelijke dialoog inprogrammeren zodat het inloggen automatisch gaat.

Transfer text/binary files

Tekst- files kunnen alle Kermit-implementaties uitwisselen, daar zijn ze tenslotte voor gemaakt. Binaire files kunnen de meeste Kermits ook uitwisselen. Hiermee kan men bijvoorbeeld complete executables verzenden. Natuurlijk heeft het uitwisselen van executables alleen zin als de systemen hetzelfde zijn. Ook gecomprimeerde .ARC files zijn binair.

Voor het uitwisselen van files wordt door Kermit gebruik gemaakt van het door Gert Klein beschreven Kermit protocol. Dit protocol is zodanig van opzet, dat de twee Kermits aan het begin van de transfer afspreken welke van de hier besproken faciliteiten wel en welke niet gebruikt zullen worden. De bediener van de local Kermit kan hoogstens zijn voorkeur kenbaar maken doch als de remote kermit de gekozen faciliteit niet ondersteunt, dan wordt deze ook niet gebruikt.

Wildcard send

Dit houdt in dat men meerdere files die verzonden moeten worden aan kan geven door middel van zogenaamde wildcards. Een wildcard is een teken dat in de plaats staat voor een ander teken of andere tekens. Een paar voorbeelden:

A*.*	De ster betekent elk teken of reeks van tekens. In dit geval dus alle files die met A beginnen.
ABC??.PAS	Het vraagteken kan vervangen worden door elk teken doch door slechts een teken. In dit geval dus alle .PAS files van vijf letters, beginnend met ABC.

File transfer interruption

Men kan het overzenden van files onderbreken en op een nette manier afsluiten.

Filename collission avoidance

Onder Kermit wordt de filenaam meegezonden. In normale gevallen wordt de file onder dezelfde naam op het andere systeem opgeslagen voor zover de filenaam voldoet aan de conventies van het operating systeem. Men kan overigens aangeven dat de file een andere naam moet krijgen. Nu kan het dus gebeuren dat er een file overgestuurd wordt die dezelfde naam heeft als een andere file die al op het andere systeem aanwezig is. Kermit zorgt er dan voor dat de file een andere naam krijgt zodat de reeds aanwezige file niet overschreven wordt.

Can time out

Als er op een of andere manier iets fout gaat bij de filetransfer, is het handig als het systeem niet tot Sint Juttemis blijft wachten maar probeert de uitwisseling weer op gang te brengen en als dit niet gaat, stopt met een foutboodschap. Deze optie is in vrijwel alle Kermits aanwezig waarbij, bij de meer geavanceerde implementaties, de maximale wachttijd (time out) vaak instelbaar is.

8th-bit prefixing, Repeat Count prefixing

Zoals in het artikel van Gert Klein als is aangegeven, kunnen binaire files ook met het Kermit-protocol overgestuurd worden. Het protocol schrijft voor, dat alle stuurtekens door middel van een zogenaamde prefix aangegeven worden. cntrl-A (SOH) wordt aangegeven met #A. Ingeval van data waarbij het hoogste bit geset is, zijn er twee mogelijkheden: 1) Er wordt uitsluitend 7-bits data overgestuurd en 2) er wordt 8-bits data overgestuurd. Als er 7-bits data overgestuurd wordt, dan wordt er door middel van een prefix (in dit geval de &) aangegeven dat in het databyte bit 8 geset moet worden.

Repeat Count prefixing kan gebruikt worden om meerdere van dezelfde tekens achter elkaar op een korte manier aan te geven. Men kan op deze manier tot maximaal 94 van dezelfde tekens in 3 tekens aangeven. De onderstaande voorbeelden komen uit de Kermit Protocol Manual:

^ betekent Cntrl
´ betekent 8e bit geset
"(" is ASCII 40 - 32 = 8

Teken	Representatie met Prefix(es)	Met herhaling voor 8 tekens
A (41)	A	~(A
^A (01)	#A	~(#A
´A (C1)	&A	~(&A
´^A (81)	&#A	~(&#A

Kan men over de lijn 8 bit data sturen, dan zien de bovenstaande voorbeelden er als volgt uit:

Teken	Representatie met Prefix(es)	Met herhaling voor 8 tekens
A (41)	A	~(A
^A (01)	#A	~(#A
´A (C1)	´A	~(´A
´^A (81)	#´A	~(#´A

Noot ´ is dus geen prefix; dit betekent alleen dat er een byte met het 8e bit geset overgestuurd wordt.

Wordt er in de file een prefix-teken verstuurd, dan wordt dit teken vooraf gegaan door het # teken. Dus:

Teken:	Vestuurd:
#	##
&	#&
~	#~

ujl
Het verstuurde teken kan dan nog vooraf worden gegeaan door de 8th-bit prefix en de repeat count prefix.

Alternate Block Checks

Alle Kermit-implementaties ondersteunen op zijn minst de meest eenvoudige vorm van Block Check. Deze vorm is in het artikel van Gert Klein besproken. Een aantal Kermit-implementaties kennen verder nog twee andere vormen. De eerste is gebaseerd op een twaalf bits checksum; de tweede op de 16-bit CRC-CCITT die ook in de meeste netwerken gebruikt wordt. In het algemeen voldoet de eenvoudige vorm uitstekend. Heeft men echter binaire files die over slechte verbindingen verstuurd worden, dan is een alternatieve block check mogelijk iets beter, omdat eventuele fouten dan beter te detecteren zijn.

Terminal emulation

In een voorgaande paragraaf is al aangege-

ven dat men het Kermit-programma kan gebruiken om net zoals met een terminal op een remote systeem te kunnen werken. Om dit goed te kunnen doen, bevatten de meeste Kermits een een of meer terminal emulaties. In bijna alle gevallen is dat minstens een DEC VT52 emulatie. Dit betekent dat het voor het remote systeem net is alsof er een terminal van het geemuleerde type een verbinding heeft. Dit uiteraard niet tijdens de file-transfer maar in de zogenaamde Connect mode.

Communication Settings, Transmit Break

Bij een groot aantal Kermit-programma´s is het mogelijk de parameters voor de communicatie in te stellen (Baud-rate, parity, word-length ...). Verder zijn er een aantal implementaties waarmee een BREAK-signaal verzonden kan worden. Dit signaal kan voor het remote-systeem bijvoorbeeld betekenen dat het sturen van tekens afgebroken moet worden (de Junior reageert bijvoorbeeld op deze manier op een break).

Support voor dialout modems

Zoals bekend, zijn er een groot aantal modems die zelf een telefoonnummer kunnen kiezen en van spraak naar data kunnen schakelen als er verbinding verkregen wordt. De nieuwere Kermit-implementaties ondersteunen deze faciliteit voor een aantal modems. Men kan dan dus simpelweg het commando DIAL 053,303902 opgeven.

IBM mainframe communication

Het communiceren met IBM-mainframes vraagt extra inspanning voor het bedienen van het communicatiekanaal (Half Duplex bijvoorbeeld). Een aantal Kermit-versies zijn in staat deze extra inspanning te leveren.

Transaction, Session, Debug and Packet logging

Kermit is in staat informatie over een aantal zaken in een file weg te schrijven. De hoeveelheid informatie is afhankelijk van de soort logging die geactiveerd wordt. Met session logging kan men alles wat in connect-toestand (terminal-emulatie) verzonden en ontvangen wordt vastleggen. Op deze manier haal ik bijvoorbeeld informatie van mijn Junior systeem zonder Kermit over naar mijn andere systemen. Ik start de session logging en geef een "List" commando op de Junior. Na het listen van de file, sluit ik de logfile, haal de overbodige informatie er met de editor uit en zie daar, ik heb de (tekst-) file op mijn andere systeem. Debug logging geeft een overzicht van alle toestanden die kermit doormaakt tijdens het draaien van het programma. Vooral bij file-transfer is het interessant dit eens te bestuderen. Het is vooral bedoeld om een nieuwe Kermit-implementatie snel foutvrij te maken.

Act as Server, Talk to Server

De meer geavanceerde versies van Kermit kennen de zogenaamde Server commando´s. Deze zijn vooral bedoeld voor file transfer. Een van de twee Kermits, meestal de remote Kermit, fungeert dan als server. Op de locale Kermit kan men dan ingeven wat men van de server verwacht; het opsturen van files (uploaden) of het ontvangen van files (downloaden). De locale Kermit geeft deze opdracht dan door aan de server die het commando uitvoert. Men hoeft dan slechts een van de twee systemen te vertellen wat men wil.

Heeft men geen server-mode, dan moet men op het zendende systeem een "Send" commando ingeven en op het ontvangende systeem een "Receive" commando. De meeste Kermit-implementaties kunnen met een server communiceren, een groot aantal (o.a. DOS-65 Kermit) kunnen ook als server optreden. Als gebruiker van Kermit vindt ik dit het sterkste punt van het pakket.

Advanced server functions

Ook in de servers kent men eenvoudige en meer geavanceerde systemen. De eenvoudige systemen kunnen alleen files zenden en ontvangen, de meer geavanceerde systemen kunnen bovendien onder andere directorie-listings doorsturen. Er bestaan implementaties waarbij men aan de server alle commando´s van het operating systeem waaronder de server draait kan geven.

Local file management

Binnen de Kermits met deze optie kan men van directory wisselen, een directory opvragen files verwijderen etc. zonder uit het Kermit programma te gaan.

Command/Init files

Met deze optie kunnen commando´s aan Kermit in een file samengevoegd worden en door middel van een simpele aanroep of automatisch bij het starten uitgevoerd worden. Met een dergelijke file en het script commando kan men bijvoorbeeld:

- De communicatiepoort initialiseren
- D.m.v. DIAL het nummer van een BBS

draaien
- D.m.v. Script inloggen
- Overgaan naar Connect

UUCP and multiuser line locking

Deze optie is bedoeld om op een UNIX systeem ervoor te zorgen dat niet meerdere gebruikers gelijktijdig via dezelfde communicatielijn communiceren. (UUCP = Unix to Unix Copy program).

Long packets

De meeste Kermits gebruiken in de file-transfer een packet-lengte van maximaal 95 bytes. Er zijn implementaties waarbij dit uitgebreid kan worden tot bijvoorbeeld 1024 bytes. Hoe langer het packet, hoe kleiner het deel overhead voor de stuurtekens aan het begin en het eind van het packet en hoe minder vaak er gewacht hoeft te worden op de bevestiging. Lange packets hebben echter als nadeel dat bij een geconstaerde fout men meer tekens opnieuw over moet sturen.

Sliding windows

Bij de communicatie via satellieten of met het BBS op Mars, duurt het enige tijd voordat na het verzenden van het laatste teken van een packet, de bevestiging hierop binnenkomt. De informatie wordt namelijk met een snelheid van 300.000 km per seconde getransporteerd. Naar Mars kan dit 40 minuten duren. Om dit te ondervangen is in het Kermit-protocol voorzien dat er meerdere packets onderweg gestuurd worden alvorens er op de bevesting van de eerste gewacht wordt. In normale omstandigheden is deze uitbreiding niet echt van belang.

File attribute packets

Op de meeste systemen wordt bijgehouden van wie een file is en wat anderen met deze file mogen doen. Bovendien wordt meestal de datum en tijd van het moment waarop de file aangemaakt is bewaard. Er zijn Kermit-implementaties die in staat zijn deze informatie ook op te sturen en weer te verwerken in de ontvangen files.

Command Macro´s

Met een Command Macro kan men nieuwe commando´s definieren die samengesteld zijn uit andere commando´s. Op deze manier kan men dus de commando´s voor Kermit tijdelijk uitbreiden. Voorbeeld:

```
DEFINE IBM =  PARITY ODD,  DUPLEX HALF,  \
              HANDSHAKE XON
```

Door middel van SET IBM kan men dan in een klap de genoemde settings instellen.

Raw file transmit

Bij Raw file transmit, kan men files ontvangen van systemen die geen Kermit hebben. In de praktijk betekent dit dat men op het locale systeem een file opent en alle informatie die binnenkomt daar inschrijft. Op het remote systeem doet men dan een List. Op deze manier wordt er dus geen enkele controle op het al dan niet goed ontvangen van de informatie uitgevoerd.

Afsluiting

Natuurlijk is er over Kermit nog aanzienlijk meer te vertellen. Na de mogelijkheden is een commando-overzicht van een gemiddelde Kermit-implementatie een mogelijk vervolg. Misschien dat iets dergelijks in de toekomst ook nog eens geplaatst gaat worden. Vooralsnog wil ik het hierbij even laten. Ik hoop dat duidelijk geworden is dat, ondanks het feit dat Kermit vrij beschikbaar is, dit toch een programma is dat professionele allures heeft. Wil men meer informatie, dan zijn er een tweetal publicaties beschikbaar:

1) Frank da Cruz: KERMIT PROTOCOL MANUAL, Sixth edition June 1986
2) Frank da Cruz, Editor: KERMIT USER GUIDE, Sixth edtion, Revision 2, May 26 1986

Verder natuurlijk de user guide voor uw eigen systeem o.a. de KERMIT USER GUIDE voor DOS-65 of de UNIX KERMIT USER GUIDE in de versie van januari 1988. Al deze documentatie kan eventueel via de redactie verkregen worden. Hebt u een systeem en wilt u weten of daar een Kermit voor beschikbaar is, dan kunt u het beste contact opnemen met Gert Klein.

==========

Othello

Het onderstaande programma is reeds enige jaren oud. Ik denk toch dat het programma nog steeds interessant kan zijn. Aanleiding tot het publiceren van het programma vormde het volgende bericht uit de Rijn en Gouwe van zaterdag 16 juli 1988:

Olympiade voor computer games

Volgend jaar augustus wordt in het Londense Park Hotel 's werelds eerste Olympiade voor spelende, kunstmatig intelligente computers gehouden. Organisator is de bekende schaakgrootmeester en programmeur David Levy. De wedstrijden zijn alleen toegankelijk voor computers of spelmachines op basis van microprocessors.
Onderdelen van de Olympiade kunnen alle door machines tegen machines te spelen strategische spelen zijn: schaken, dammen, backgammon, bridge, go moku, scrabble, mancala, cribbage, **Othello**, stratego, Go ban, May Yong, enzovoorts. Ook zullen twee nieuwe strategische computergames worden geïntroduceerd: het Japanse Shogi en het Chinese Xianggi.
Mensen zijn van deelneming uitgesloten. Ze mogen hooguit dienst doen als toeschouwers of als robot voor het invoeren van de zetten. De arbitrage zal aan mensen worden opgedragen.
Tijdens de Londense Computer Olympiade wordt een groot symposium gehouden over kunstmatige intelligentie en de toekomst van de schaak- en spelcomputer.

In HCC-nieuwsbrief nr. 61 stond een programma voor het spel Othello afgedrukt. Omdat ik in die tijd experimenteerde met Comal, heb ik dit programma geconverteerd naar de door de club verspreide COMAL.KGN. Deze versie van het programma liep op Junior met Elekterminal. Door enkele kleine modificaties kan het programma ook lopen op andere systemen waarop de genoemde (of een andere?) Comal draait. Interessanter is waarschijlijk het programma nogmaals te converteren naar 'C' of Pascal en het dan strategisch behoorlijk te versterken. Misschien komen we dan nog zo ver dat een of meer van de leden van de club meedoet aan de bovengenoemde Olympiade.....

```
10  //    *** OTHELLO OF REVERSI ***
20  //    *** VERSIE 1  10-09-84 ***
30  //
40  //    BEWERKING VAN HET PROGRAMMA VAN S. NIJHUIS
50  //    UIT DE HCC-NIEUWSBRIEF 61.
60  //
70  //    AUTEUR: GERT VAN OPBROEK   (GEVOP)
80  //                  BATEWEG 60
90  //                  2481 AN WOUBRUGGE
100  //                 01729-8636
110  //
120  //    HET PRGRAMMA WERD ONTWIKKELD OP EEN SENIOR MET
130  //    ELEKTERMINAL.
140  //
150  //    VOOR GEBRUIK MET EEN ANDERE TERMINAL MOETEN DE
160  //    PROCEDURES "CLS", "HOME" EN "C_POS" AANGEPAST
170  //    WORDEN.
180  //
190  DIM  A(8,8),CH$(2),SO(2)
200 CH$(0) := " ";CH$(1) := "*";CH$(2) := "#"
210  EXEC:  "REGELS"
220  EXEC:  "VOORBEELD"
230  EXEC:  "SPEL"
240 VE := 13;HO := 0
```

```
250  EXEC:  "C_POS",VE,HO
260  PRINT  "EINDE........."
270  END.
271   //
280  PROC  "CLS"    //   CLEAR-SCREEN OF ELEKTERMINAL
290   POKE  6745,200
300   PRINT   CHR$ (12);
310   POKE  6745,2
320  ENDPROC
330  //
340  PROC  "HOME"    //   CURSOR HOME ON ELEKTERMINAL
350   POKE  6745,200
360   PRINT   CHR$ (28);
370   POKE  6745,2
380  ENDPROC
390  //
400  PROC  "C_POS",VE,HO    //   MOVE CURSOR TO POSITION VE,HO
410   EXEC:  "HOME"
420   PRINT
430   IF  HO > 0 THEN
440    FOR  OD := 1  TO  HO
450     PRINT   CHR$ (9);
460    ENDFOR
470   ENDIF
480   FOR  OF :=  - 1  TO  15 - VE
490    PRINT   CHR$ (11);
500   ENDFOR
510  ENDPROC
520  //
530  PROC  "KOP"    //   GEEF EEN KOP
540   EXEC:  "CLS"
550   PRINT    SPC( 18);"*** OTHELLO OF REVERSI ***"
560   PRINT    SPC( 18);"=========================="
570  ENDPROC
580  //
590  PROC  "VELD"    //   TEKEN HET SPEELVELD
600   EXEC:  "KOP"
610   PRINT    SPC( 10);"1  2  3  4  5  6  7    "
620   PRINT    SPC( 10);"++ ++ ++ ++ ++ ++ ++ ++"
630   FOR  I := 1  TO  8
640    PRINT    SPC( 7);I;"+ "
650   ENDFOR
660  ENDPROC
670  //
680  PROC  "ZET",RY,KO,SI    //   VOER ZET:RY,KO UIT VOOR SPELER SI
690  A(RY,KO) := SI
700  HO := 7 + 3 * KO;VE := 3 + RY
710   EXEC:  "C_POS",VE,HO
720   PRINT  CH$(SI);CH$(SI);
730  ENDPROC
740  //
750  PROC  "TEL"    //   TEL DE STEENVERDELING EN DRUK DEZE AF
760   FOR  I := 0 TO  2
770   SO(I) := 0
780   ENDFOR
790   FOR  I := 1  TO  8
800    FOR  J := 1  TO  8
810    SO(A(I,J)) := SO(A(I,J)) + 1
820    ENDFOR
830   ENDFOR
840  HO := 40;VE := 4
850   EXEC:  "C_POS",VE,HO
860   PRINT  "SPELER 1 (";CH$(1);") :";SO(1);" ";
870  HO := 40;VE := 5
```

```
880   EXEC:  "C_POS",VE,HO
890   PRINT  "SPELER 2 (";CH$(2);") :";SO(2);" ";
900  ENDPROC
910  //
920  PROC  "INIT"
930   FOR  I := 1  TO  8
940    FOR  J := 1  TO  8
950    A(I,J) := 0
960    ENDFOR
970   ENDFOR
980   EXEC:  "VELD"
990  RY := 4;KO := 4;SI := 1
1000   EXEC:  "ZET",RY,KO,SI
1010  RY := 5;KO := 5
1020   EXEC:  "ZET",RY,KO,SI
1030  KO := 4;SI := 2
1040   EXEC:  "ZET",RY,KO,SI
1050  RY := 4;KO := 5
1060   EXEC:  "ZET",RY,KO,SI
1070   EXEC:  "TEL"
1080  ENDPROC
1090  //
1100  PROC  "TESTBUUR",RY,KO,TY,RE
1110   //   TEST OP DE AANWEZIGHEID VAN BUREN VAN TYPE TY
1120   //   HET RESULTAAT KOMT IN RE
1130  RE := 0;I := RY - 2
1140   WHILE  (I < RY + 1)  AND  (RE = 0)  DO:
1150   I := I + 1;J := KO - 2
1160    WHILE  (J < KO + 1)  AND  (RE = 0)  DO:
1170    J := J + 1
1180     IF  (I > 0)  AND  (I < 9)  AND  (J > 0)  AND  (J < 9)  THEN
1190      IF  A(I,J) = TY  THEN
1200      RE := 1
1210      ENDIF
1220     ENDIF
1230    ENDWHILE
1240   ENDWHILE
1250  ENDPROC    //   TESTBUUR
1260  //
1270  PROC  "TELBUUR",XX,YY,TT,G,T
1280   //   TELT HET AANTAL TE PAKKEN STENEN VAN KLEUR G
1290   //   INDIEN T=0 WORDEN DE ZETTEN UITGEVOERD.
1300  TT := 0;TY := 3 - G
1310   FOR  SX :=  - 1  TO  1
1320    FOR  SY :=  - 1  TO  1
1330     IF  (SX <  > 0)  OR  (SY <  > 0)  THEN
1340     I := XX;J := YY
1350     IK := I + SX;JK := J + SY
1360      IF  (IK > 0)  AND  (IK < 9)  AND  (JK > 0)  AND  (JK < 9)  THEN
1370       REPEAT
1380       I := I + SX;J := J + SY
1390       IK := I + SX;JK := J + SY
1400       UNTIL  (IK < 1)  OR  (IK > 8)  OR  (JK < 1)  OR  (JK > 8)  OR  (A(I,J) <  > TY)
1410      ENDIF
1420      IF  (A(I,J) = G)  AND  (( ABS (I - XX) > 1)  OR  ( ABS (J - YY) > 1))  THEN
1430      I := I - SX;J := J - SY
1440       WHILE  (I <  > XX)  OR  (J <  > YY)  DO:
1450       TT := TT + 1
1460        IF  T = 0 THEN
1470         EXEC:  "ZET",I,J,G
1480        ENDIF
1490       I := I - SX;J := J - SY
1500       ENDWHILE
1510      ENDIF
```

```
1520     ENDIF
1530    ENDFOR
1540   ENDFOR
1550  ENDPROC    //   TELBUUR
1560  //
1570  PROC  "REGELS"
1580   //   GEEFT (INDIEN NODIG) DE SPELREGELS.
1590   EXEC:  "KOP"
1600  HO := 0;VE := 12
1610   EXEC:  "C_POS",VE,HO
1620   PRINT  "WILT U SPELREGELS? (N/J)";
1630   REPEAT
1640    PRINT  "?";
1650    GET  A$
1660   UNTIL  (A$ = "N")  OR  (A$ = "J")
1670   EXEC:  "C_POS",VE,HO
1680   PRINT    SPC( 60);
1690   IF  A$ = "J"  THEN
1700   VE := 3;HO := 0
1710    EXEC:  "C_POS",VE,HO
1720    PRINT  "HET GAAT ER BIJ DIT SPEL OM DE MEESTE VELDEN TE BEZETTEN."
1730    PRINT  "EEN VELD WORDT BEZET DOOR ER EEN ´STEEN´ OP TE PLAATSEN,"
1740    PRINT  "ZODANIG DAT ER EEN OF MEERDERE VELDEN VAN DE TEGENPARTIJ"
1750    PRINT  "WORDEN INGESLOTEN. DIT KAN ZOWEL HORIZONTAAL ALS VERTICAAL"
1760    PRINT  "EN DIAGONAAL. ER ZIJN 64 VELDEN EN HET SPEL STOPT ALS ALLE"
1770    PRINT  "VELDEN BEZET ZIJN, OF ALS 1 VAN DE SPELERS GEEN VELD MEER"
1780    PRINT  "IN BEZIT HEEFT."
1790    PRINT
1800    PRINT  "U SPEELT MET DE ";CH$(1);" EN DE COMPUTER MET DE ";CH$(2);"."
1810    PRINT
1820    PRINT  "ALS U GEEN VELD KUNT INSLUITEN, MOET U EEN BEURT OVERSLAAN"
1830    PRINT  "DOOR 9,9 TE ZETTEN."
1840    PRINT  "<<< GEEF EEN RETURN>>>";
1850    GET  A$
1860   ENDIF
1870  ENDPROC
1880  //
1890  PROC  "HULP",S1$,S2$,X,Y,G
1900   //   HULPROUTINE VOOR VOORBEELDEN
1910  VE := 12;HO := 0
1920   EXEC:  "C_POS",VE,HO
1930   PRINT    SPC( 60)
1940   PRINT    SPC( 60)
1950   PRINT    SPC( 60)
1960   IF  (X > 0)  AND  (Y > 0)  AND  (G > 0)  THEN
1970   T := 0
1980    EXEC:  "ZET",X,Y,G
1990    EXEC:  "TELBUUR",X,Y,TT,G,T
2000    EXEC:  "TEL"
2010   ENDIF
2020  HO := 0;VE := 12
2030   EXEC:  "C_POS",VE,HO
2040   PRINT  S1$
2050   PRINT  S2$
2060   PRINT  "<<< GEEF EEN RETURN >>>";
2070   GET  A$
2080  ENDPROC
2090  //
2100  PROC  "VOORBEELD"
2110   //   GEEFT (INDIEN NODIG) SPELVOORBEELDEN.
2120   EXEC:  "KOP"
2130  HO := 0;VE := 12
2140   EXEC:  "C_POS",VE,HO
2150   PRINT  "WILT U SPELVOORBEELDEN? (N/J)";
```

```
2160   REPEAT
2170    PRINT  "?";
2180    GET  A$
2190   UNTIL  (A$ = "N")  OR  (A$ = "J")
2200   EXEC:  "C_POS",VE,HO
2210   PRINT    SPC( 60);
2220   IF  A$ = "J"  THEN
2230    EXEC:  "INIT"
2240   S1$ := "DIT IS HET VELD EN DE BEGINSTAND."
2250   S2$ := CH$(2) + " IS AAN ZET EN DOET: 4,3"
2260   X := 0;Y := 0;G := 0
2270    EXEC:  "HULP",S1$,S2$,X,Y,G
2280   X := 4;Y := 3;G := 2
2290   S1$ := "HET ANTWOORD VAN " + CH$(1) + " IS: 3,3"
2300   S2$ := ""
2310    EXEC:  "HULP",S1$,S2$,X,Y,G
2320   X := 3;Y := 3;G := 1
2330   S1$ := CH$(2) + " DOET NU 3,4"
2340    EXEC:  "HULP",S1$,S2$,X,Y,G
2350   X := 3;Y := 4;G := 2
2360   S1$ := CH$(1) + " DOET NU: 3,5....."
2370    EXEC:  "HULP",S1$,S2$,X,Y,G
2380   X := 3;Y := 5;G := 1
2390   S1$ := "EN PAKT HIERMEE 2(!) STENEN."
2400   S2$ := "ENZ. ENZ. TOT HET VELD VOL IS; SUCCES."
2410    EXEC:  "HULP",S1$,S2$,X,Y,G
2420   ENDIF
2430  ENDPROC
2440  //
2450  PROC  "INVOER",RF
2460   //   VOER EEN ZET IN EN VOER HEM UIT
2470   REPEAT
2480   RF :=  - 1
2490   HO := 0;VE := 12
2500    EXEC:  "C_POS",VE,HO
2510    PRINT  "10,1=OPGEVEN; 9,9=U KUNT GEEN VELD INSLUITEN."
2520    PRINT  "GEEF VAN U ZET DE RY,KOLOM GESCHEIDEN DOOR EEN KOMMA.";
2530    INPUT  AN$
2540   I := 0
2550    WHILE  (I <  LEN (AN$))  AND  ( MID$ (AN$,I,1) <  > ",")  DO:
2560    I := I + 1
2570    ENDWHILE
2580   RY :=  VAL ( LEFT$ (AN$,I));KO :=  VAL ( RIGHT$ (AN$, LEN (AN$) - I))
2590   K1 := (RY > 0)  AND  (KO > 0)  AND  (RY < 9)  AND  (KO < 9)
2600   K1 := K1  OR  ((RY = 9)  AND  (KO = 9))  OR  ((RY = 10)  AND  (KO = 1))
2610    IF    NOT  (K1)  THEN
2620     PRINT  "PARDON?";
2630    ELSE
2640    HO := 0;VE := 12
2650     EXEC:  "C_POS",VE,HO
2660     PRINT    SPC( 60)
2670     PRINT    SPC( 60)
2680     PRINT    SPC( 60);
2690     IF  RY < 9  THEN
2700      IF  A(RY,KO) = 0 THEN
2710      T := 0;G := 1
2720       EXEC:  "ZET",RY,KO,G
2730       EXEC:  "TELBUUR",RY,KO,TT,G,T
2740       IF  TT = 0 THEN
2750       G := 0
2760        EXEC:  "ZET",RY,KO,G
2770       ELSE
2780        EXEC:  "TEL"
2790       RF := 0
```

```
2800       ENDIF
2810      ENDIF
2820     ELSE
2830      IF  RY = 10 THEN
2840      RF := 2
2850      ELSE
2860      RF := 1
2870      ENDIF
2880     ENDIF
2890    ENDIF
2900    IF  RF < 0 THEN
2910    HO := 0;VE := 14
2920     EXEC:  "C_POS",VE,HO
2930     PRINT  "UW ZET WAS ONGELDIG; OPNIEUW.";
2940    ENDIF
2950   UNTIL  RF >  = 0
2960  ENDPROC
2970  //
2980  PROC  "NIVEAU2",RY,KO,TT,T1,T2,VE,HO,VL
2990   //
3000  KL := (RY = 1)  AND  (KO = 2)  AND  (A(1,1) <  > 2)
3010  KL := KL  OR  (RY = 1)  AND  (KO = 7)  AND  (A(1,8) <  > 2)
3020  KL := KL  OR  (RY = 2)  AND  ((KO = 1)  OR  (KO = 2))  AND  (A(1,1) <  > 2)
3030  KL := KL  OR  (RY = 2)  AND  ((KO = 7)  OR  (KO = 8))  AND  (A(1,8) <  > 2)
3040  KL := KL  OR  (RY = 7)  AND  ((KO = 1)  OR  (KO = 2))  AND  (A(8,1) <  > 2)
3050  KL := KL  OR  (RY = 7)  AND  ((KO = 7)  OR  (KO = 8))  AND  (A(8,8) <  > 2)
3060  KL := KL  OR  (RY = 8)  AND  (KO = 2)  AND  (A(8,1) <  > 2)
3070  KL := KL  OR  (RY = 8)  AND  (KO = 7)  AND  (A(8,8) <  > 2)
3080   IF  KL  THEN
3090    IF  (TT > T2)  AND  (T1 < .6)  THEN
3100    T1 := .5;VE := RY;T2 := TT;HO := KO
3110    VL := 0
3120    ENDIF
3130   ENDIF
3140  ENDPROC    //   "NIVEAU2"
3150  //
3160  PROC  "NIVEAU3",RY,KO,TT
3170   //
3180  KK :=   NOT  ((KO = 2)  OR  (KO = 7))
3190  KR :=   NOT  ((RY = 2)  OR  (RY = 7))
3200  KL := ((RY = 1)  OR  (RY = 8))  AND  KK
3210  KL := KL  OR  ((KO = 1)  OR  (KO = 8))  AND  KR
3220   IF  KL  THEN
3230   TT := TT + 3
3240   ENDIF
3250  ENDPROC    //   "NIVEAU3"
3260  //
3270  PROC  "NIVEAU4",RY,KO,TT
3280   //
3290  KK := (KO = 2)  OR  (KO = 7)  OR  (KO = 1)  OR  (KO = 8)
3300  KR := (RY = 2)  OR  (RY = 7)  OR  (RY = 1)  OR  (RY = 8)
3310  KL := ((RY = 2)  OR  (RY = 7))  AND    NOT  KK
3320  KL := KL  OR  ((KO = 2)  OR  (KO = 7))  AND    NOT  KR
3330   IF  KL  THEN
3340   TT := TT - .4
3350   ENDIF
3360  ENDPROC    //   "NIVEAU4"
3370  //
3380  PROC  "SPEL"
3390   EXEC:  "KOP"
3400  HO := 0;VE := 12
3410   EXEC:  "C_POS",VE,HO
3420   PRINT  "GEEF DE MOEILIJKHEIDSGRAAD. (1..4)";
3430   REPEAT
```

```
3440    GET  M$
3450    IF  (M$ < "1")  OR  (M$ > "4")  THEN
3460     PRINT  "?";
3470    ENDIF
3480   UNTIL  (M$ > "0")  AND  (M$ < "5")
3490   EXEC:  "INIT"
3500   REPEAT
3510    EXEC:  "INVOER",RF
3520    IF  RF < 2  THEN
3530    G := 2;TY := 1;T2 := 0;T1 := 0
3540     FOR  RY := 8  TO  1  STEP  - 1
3550      FOR  KO := 8  TO  1  STEP  - 1
3560       IF  A(RY,KO) = 0 THEN
3570        EXEC:  "TESTBUUR",RY,KO,TY,RE
3580        IF  RE = 1  THEN
3590        T := 3;TT := 0
3600         EXEC:  "TELBUUR",RY,KO,TT,G,T
3610         IF  TT <  > 0 THEN
3620         VL := 1
3630          IF  M$ = "3"  OR  M$ = "4"  THEN
3640           EXEC:  "NIVEAU3",RY,KO,TT
3650          ENDIF
3660          IF  (TT <  = 6)  AND  (TT > T1)  AND  (M$ <  > "1")  THEN
3670           IF  M$ = "4"  THEN
3680            EXEC:  "NIVEAU3",RY,KO,TT
3690           ENDIF
3700           EXEC:  "NIVEAU2",RY,KO,TT,T1,T2,VE,HO,VL
3710          ENDIF
3720          IF  (M$ < "3")  AND  (VL > 0)  THEN
3730          KL := (RY = 1)  AND  ((KO = 1)  OR  (KO = 8))
3740          KL := KL  OR  ((RY = 8)  AND  ((KO = 1)  OR  (KO = 8)))
3750           IF  KL  THEN
3760           TT := TT + 6
3770           ENDIF
3780          ENDIF
3790          IF  (VL > 0)  AND  (TT > T1)  THEN
3800          T1 := TT;VE := RY;HO := KO
3810          ENDIF
3820         ENDIF
3830        ENDIF
3840       ENDIF
3850      ENDFOR
3860     ENDFOR
3870    RY := VE;KO := HO;T := 0
3880     IF  T1 > 0 THEN
3890      EXEC:  "ZET",RY,KO,G
3900      EXEC:  "TELBUUR",RY,KO,TT,G,T
3910      EXEC:  "TEL"
3920      IF  (SO(1) = 0)  OR  (SO(2) = 0)  OR  (SO(0) = 0)  THEN
3930      RF := 2
3940      ENDIF
3950     ELSE
3960      IF  RF = 1  THEN
3970      RF := 2
3980      ENDIF
3990     ENDIF
4000    ENDIF
4010   UNTIL  RF = 2
4020  ENDPROC
```

TECHNITRON TLP-12 LASER PRINTER
— U HEEFT EIGENLIJK GEEN ANDERE KEUZE!

- 12 pagina's per minuut (max.)
- tot 10.000 afdrukken per maand
- 8 ingebouwde lettertypes; 32 afdruk-combinaties
- unieke "FontMaker" service
- unieke "FormsMaker", formulier- en logo service
- 3 ingebouwde hardware-emulaties
- flexibele in- en uitvoer van papier

Technitron DATA

Technitron Data B.V.
Zwarteweg 110, Postbus 14,
1430 AA Aalsmeer
tel. 02977-22456
telefax 02977-40968
telex 13301

Vestigingen in:
BONDSREPUBLIEK DUITSLAND – DENEMARKEN – ENGELAND – FRANKRIJK – ITALIË – NOORWEGEN – VERENIGDE STATEN – ZWEDEN